John K. Kruschke

Doing Bayesian Data Analysis

A Tutorial with R and BUGS

parameter values are negatively correlated, such that when we believe one parameter value is large, we believe that the other parameter value is small. That negative correlation causes the conjoint distribution to straddle the line of equality.

In summary, the marginal distributions of single parameters do not indicate the relationship between the parameter values. The conjoint distribution of the two parameters might have positive or negative correlation (or even a non-linear dependency), and therefore the difference of the parameter values should be explicitly examined.

12.1.3 Region of Practical Equivalence (ROPE)

The estimation approach can be enhanced by including a region of practical equivalence (ROPE), which indicates a small range of values that are considered to be practically equivalent to the null value for purposes of the particular application. For example, if we wonder whether a coin is fair, for purposes of determining which team will kick off at a football game, then we want to know if the underlying bias in the coin is reasonably close to 0.50, and we don't really care if the true bias is 0.51 or 0.49, because those values are close enough for our application. As another example, if we are assessing the efficacy of a drug versus a placebo, we might only consider using the drug if it improves the probability of cure by at least three percentage points, and we would reject the drug to be counterproductive if it decreases the probability of cure at all, i.e., the lower boundary of the ROPE would be zero difference.

Once a ROPE is set, we make decisions according the following rule:

> A parameter value is declared to be not credible, or rejected, if its entire ROPE lies outside the 95% HDI of the posterior distribution of that parameter.

For example, suppose that we want to know whether a coin is fair, and we establish a ROPE that goes from .45 to .55. We flip the coin 500 times and observe 325 heads. If the prior is uniform, the posterior has a 95% HDI from .608 to .691, which is completely outside the ROPE. Therefore we declare that the null value of 0.5 is rejected for practical purposes.

Because the ROPE and HDI can overlap in different ways, there are different decisions that can be made. In particular, we can decide to "accept" a value:

> A parameter value is declared to be accepted for practical purposes if that value's ROPE completely contains the 95% HDI of the posterior of that parameter.

For example, suppose that we want to know whether a coin is fair, and we establish a ROPE that goes from .45 to .55. We flip the coin 1,000 times and observe 490 heads. If the prior is uniform, the posterior has a 95% HDI from .459 to .521, which falls completely within the ROPE. Therefore we declare that the null value of 0.5 is accepted for practical purposes, because all of the credible values are practically equivalent to the null value.

This use of a ROPE around the null value also implies that if the null value really is true, we will eventually "accept" the null value as the sample size gets large enough. This is because, as the sample size gets larger, the HDI tends to get narrower and closer to the true value. When the sample size gets very large, the HDI is almost certain to be narrow enough, and close enough to the true value, to fall entirely within the ROPE. If we did not use a ROPE around the null value, and rejected the null value any time that it falls outside the 95% HDI, then we would incorrectly reject the null in 5% of experiments even if when null value is true. This 5% false alarm rate occurs because random samples of data,

when generated by the null value, will have coincidences that cause them to produce a 95% HDI that does not include the null value. Although 5% of those 95% HDIs exclude the null value, the HDIs do get narrower and closer to the null value as N gets large. Therefore the HDIs do eventually fall within the ROPE around the null, even though 5% of the 95% HDIs exclude the null value itself. As an example of this point: Suppose we establish a ROPE that goes from .45 to .55. We flip the coin 10,000 times and observe 5200 heads. The 95% HDI goes from .510 to .530. Despite the fact that the HDI excludes the null value, we accept the null for practical purposes because the HDI falls entirely within the ROPE, which means that all of the credible values are practically equivalent to the null value. We would say that we believe the true bias in the coin is very nearly .52, but that's close enough to the null value of .50 for practical purposes.

How is the size of the ROPE determined? If the application domain is a domain in which costs and benefits of decisions can be determined, then the full-blown theoretical machinery of expected utility can be brought to bear. In some domains such as medicine, expert clinicians can be interviewed, and their opinions can be translated into a reasonable consensus regarding how big of an effect is useful or important for the application. Otherwise, the ROPE might be established with somewhat arbitrary criteria, bearing in mind the key trade-off: When the ROPE is wider, there is a lower probability of falsely rejecting the null value (i.e., there is a smaller false alarm rate), but there is also a lower probability of rejecting the null value when it is false (i.e., there is a smaller hit rate). For further discussion of the ROPE, under somewhat different appellations of "range of equivalence" and "indifference zone", see e.g., Carlin and Louis (2009); Freedman, Lowe, and Macaskill (1984); Hobbs and Carlin (2008); Spiegelhalter, Freedman, and Parmar (1994).

It is important to be clear that any discrete declaration about rejecting or accepting a null value does not exhaustively capture our beliefs about the parameter value. Our beliefs about the parameter value are described by the full posterior distribution. When making a binary declaration, we have merely compressed all that detail into a single bit of information. The broader goal of Bayesian analysis is conveying an informative summary of the posterior, and where the value of interest falls within that posterior. Reporting the limits of an HDI region is more informative than reporting the declaration of a reject/accept decision. By reporting the HDI and other summary information about the posterior, different readers can apply different ROPEs to decide for themselves whether a parameter is practically equivalent to a null value.

12.2 The model-comparison (two-prior) approach

The previous section posed the question, is the null value among the credible values, in terms of parameter estimation. We started with an informed prior distribution on the parameters, and then examined the posterior distribution on the parameters.

Some researchers prefer instead to pose the question in terms of model comparison. In this framing of the question, the focus is not on estimating the magnitude of the parameter. Instead, the focus is on deciding which of two hypothetical priors is least unbelievable. One hypothetical prior expresses the idea that the parameter value is exactly the null value. The alternative hypothetical prior expresses the idea that the parameter could be anything (or anything but the null value). Notice that neither of these hypothetical priors is informed by prior knowledge. This lack of being informed is often taken as a desirable aspect of the approach, not a defect, because the method is thereby "objective" insofar as it obviates

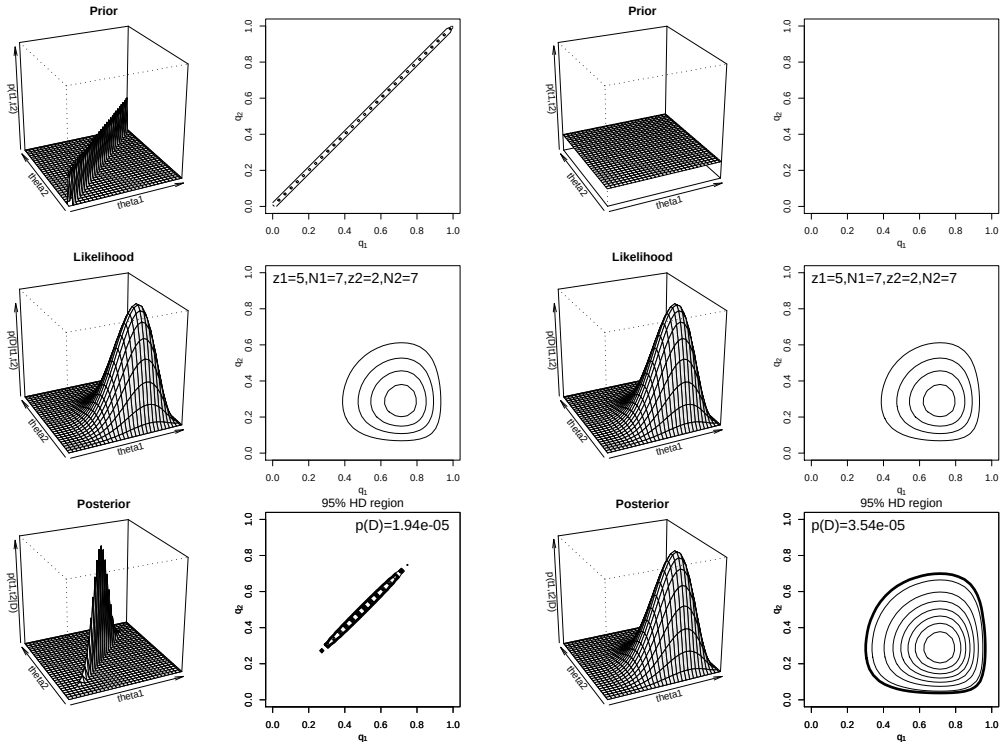

Figure 12.4: Two hypothetical priors for model comparison. The left columns show a ridge-shaped prior that allows only points for which $\theta_1 = \theta_2$. The right columns show a uniform prior, in which any combination of the values is entertained. The top-right contour plot is empty because the surface is exactly horizontal, without any height changes to mark by a contour.

disputes about prior knowledge. We will see, however, that model-comparison method can be extremely sensitive to the choice of "uninformed" prior for the alternative hypothesis (cf. Figure 10.7, p. 212), and that model comparison is not necessarily meaningful unless both models are viable in the first place.

12.2.1 Are the biases of two coins equal or not?

Consider a situation in which we have two coins, and we would like to infer whether their biases are equal or not. We pose the question as a model comparison, such that one model expresses the "null" hypothesis that the two biases are equal, and the other model expresses the "alternative" hypothesis that the two biases could be any combination. The two models are distinguished by their priors (because the likelihood functions are the same in both models): The null model has zero prior probability on all combinations of biases except where they are equal, while the alternative model has uniform probability on all combinations of biases. The uniform alternative is chosen because it is a convenient expression of non-informed indifference.

Figure 12.4 shows an example of this approach. The "null" model, M_{null}, is shown in the left columns. Notice that this prior is shaped as a ridge along all values for which $\theta_1 = \theta_2$. The prior gives zero probability to any point at which $\theta_1 \neq \theta_2$. Notice also that the height of the ridge is horizontal, meaning that it is not informed by any prior knowledge,

e.g., that the coins are probably fair. The "alternative" model, M_{alt}, is shown in the right columns of Figure 12.4. This prior puts equal probability on all possible combinations of theta values. Notice that this hypothesis also is not informed by any prior knowledge, e.g., that the coins are probably fair.

The lower row of Figure 12.4 shows the posteriors resulting from the two hypothetical priors, for the same data that have been used in previous examples such as Figure 8.1, p. 131. Notice that the posterior on the null model is a ridge like the prior, except that the extreme ends of the ridge have been attenuated because they are inconsistent with the data.

To do model comparison, we use the implication of Bayes rule:

$$\frac{p(M_{alt}|D)}{p(M_{null}|D)} = \underbrace{\frac{p(D|M_{alt})}{p(D|M_{null})}}_{BF} \frac{p(M_{alt})}{p(M_{null})} \tag{12.1}$$

where the ratio marked "BF" is the Bayes factor of the alternative model relative to the null model. The prior beliefs in each model are typically assumed to be equal, i.e. $p(M_{null}) = p(M_{alt}) = 0.5$, in the spirit of non-informed priors. The posterior plots in Figure 12.4 display the values of $p(D|M)$ for each model. The evidence for the null model is $p(D|M_{null}) = 1.94 \times 10^{-5}$, while the evidence for the alternative model is $p(D|M_{alt}) = 3.54 \times 10^{-5}$. The Bayes factor therefore slightly favors the alternative model, but not by much. Because the ratio of posterior probabilities is not very extreme, we would conclude that either model remains reasonably credible, given the data. If the Bayes factor had turned out to be more extreme, we might decide to declare one or the other prior to be less unbelievable than the other prior.

12.2.1.1 Formal analytical solution

The Bayes factor for the null and alternative models can also be computed analytically in this case. The alternative hypothesis has a uniform prior. The prior is uniform because the alternative hypothesis in this approach is supposed to be an "automatic" conventional prior that expresses a hypothesis complementary to the null hypothesis. A uniform prior on θ_1, θ_2 can be described as a product of beta distributions, namely $\text{beta}(\theta_1|1,1) \cdot \text{beta}(\theta_2|1,1)$. Therefore we can determine an exact value for $p(D|M_{alt})$ from Equation 8.5 (p. 131), which becomes

$$\begin{aligned} p(D|M_{alt}) &= \frac{B(z_1+1, N_1 - z_1+1)\, B(z_2+1, N_2 - z_2+1)}{B(1,1)B(1,1)} \\ &= B(z_1+1, N_1 - z_1+1)\, B(z_2+1, N_2 - z_2+1) \end{aligned} \tag{12.2}$$

because $B(1,1) = 1$.

We can also derive a formal analytical expression for the evidence for the null hypothesis, $p(D|M_{null})$. First, notice this: Because the null hypothesis belief is zero at any point off the ridge is zero, i.e. $p(\theta_1, \theta_2) = 0$, the double integral for the evidence simplifies to a single integral over $\theta = \theta_1 = \theta_2$:

$$\begin{aligned} p(D|M_{null}) &= \int\int d\theta_1 d\theta_2\, p(D|\theta_1, \theta_2)\, p(\theta_1, \theta_2) \\ &= \int d\theta\, p(D|\theta) \end{aligned}$$

We now plug in the Bernoulli likelihood function to get

$$\begin{aligned} p(D|M_{null}) &= \int d\theta\ p(D|\theta) \\ &= \int d\theta\ \theta^{(z_1)}(1-\theta)^{(N_1-z_1)}\theta^{(z_2)}(1-\theta)^{(N_2-z_2)} \\ &= \int d\theta\ \theta^{(z_1+z_2)}(1-\theta)^{(N_1-z_1+N_2-z_2)} \\ &= B(z_1+z_2+1,\ N_1-z_1+N_2-z_2+1) \end{aligned} \tag{12.3}$$

because, by definition, $B(a,b) \equiv \int d\theta\ \theta^{(a-1)}(1-\theta)^{(b-1)}$.

The Bayes factor for the alternative hypothesis, relative to the null hypothesis, is then the ratio of Equations 12.2 and 12.3:

$$\frac{p(D|M_{alt})}{p(D|M_{null})} = \frac{B(z_1+1,N_1-z_1+1)\,B(z_2+1,N_2-z_2+1)}{B(z_1+z_2+1,\ N_1-z_1+N_2-z_2+1)} \tag{12.4}$$

As a concrete example, consider the case presented in Figure 12.4, wherein $z_1 = 5$, $N_1 = 7$, $z_2 = 2$, and $N_2 = 7$. Plugging those values into Equation 12.4 yields $p(D|H_{alt})/p(D|H_{null}) = 1.82$. The grid approximation in Figure 12.4 displays $p(D|H_{alt}) = 3.54 \times 10^{-5}$ and $p(D|H_{null}) = 1.94 \times 10^{-5}$, yielding a ratio of 1.82, which is the same as the analytical solution.

12.2.1.2 Example application

Consider again the question of the "hot hand" in basketball, from Exercise 8.1, p. 153, which asked whether there are more successful shots after a preceding successful shot than after a preceding failed shot. For the case of one famous professional player, there were 251 successes after 285 initial successes, and 48 successes after 53 initial failures.

We can use model comparison to analyze this situation. Using Equation 12.4, we obtain a Bayes factor of 0.128, which, inverted, is a Bayes factor of 7.81 in favor of the null prior. If the priors on the models are 50:50, then the posterior probability of the null model is $p(M_{null}|D) = 88.7\%$, and the posterior probability of the alternative model is $p(M_{alt}|D) = 11.3\%$.

The model comparison suggests that the null model is less believable than the alternative model. But should we conclude, therefore, that there is zero difference between the conditions? Maybe, but not necessarily. Prior belief suggests that it is extremely unlikely that there is literally zero correlation between the first and second shots of a pair of free throws in basketball. Therefore, even though the model comparison suggests that the null prior is less unbelievable than the alternative prior, what we might really want is an estimate of the difference between success after success and success after failure. The model comparison does not provide such an estimate.

On the other hand, the direct estimation of the underlying model did provide an estimate of the difference, as was shown in Figure 12.2, p. 242. There we saw that a difference of zero was among the credible differences, but we also saw that the mean difference was a bit larger than zero, and we also saw explicitly that there is a moderately large uncertainty in the estimate of the difference. This conclusion from the direct estimate is a more informative inference from the data than the conclusion from model comparison, which states only that the ridge prior is less inconsistent with the data than the uniform prior. It should be remembered that the direct estimate of the difference began with a mildly informed prior, using

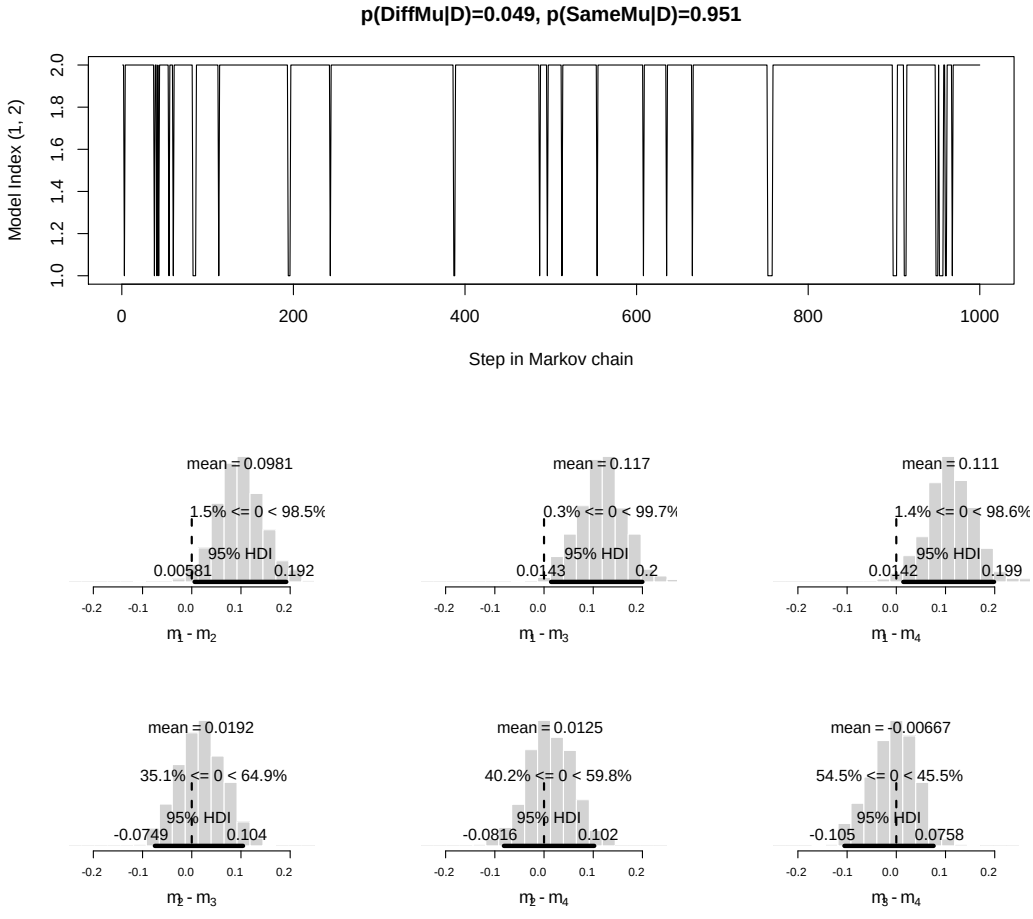

Figure 12.5: Upper Panel: Results of model comparison showing that the model with a single mu parameter ("SameMu") is preferred to a model with a separate mu parameter for each group ("DiffMu"). Histograms: Differences of posterior $_j$ values for the four groups in the different-mu model. Notice that $_1$ is credibly different from $_3$ and $_4$.

the knowledge that professional players tend to make about 75% of their free throws, unlike the the alternative model's uniform prior in the model comparison. Thus, the posterior from the alternative model is not the same as the posterior from direct estimation.

12.2.2 Are different groups equal or not?

Suppose we conduct a study that has four conditions. Suppose every participant in every condition gets the same list of twenty words to try to memorize. The ability to recall a word is modeled as a Bernoulli distribution, with probability $_{ij}$ for the i^{th} person in the j^{th} condition. The individual recall propensity $_{ij}$ depends on hyperparameters $_j$ and $_j$, that describe the overarching recall propensity in each condition, because $_{ij}$ beta($_j$ $_j$ + 1; (1 $_j$) $_j$ + 1). (Abstractly, this is a design just like the four category structures of the Ilration-condensation experiment of Section 9.3.1).

The only difference between the conditions is the type of music being played during learning and recall. For the four groups, the music comes from, respectively, the

death-metal band "Das Kruschke"[1] Mozart, Bach, and Beethoven. For the four conditions, the mean number of words recalled is 11.85, 9.85, 10.95, and 9.60. The fictitious, randomly-generated data can be examined in detail by running the program in Section 12.4.1 (OneOddGroupModelComp.R).

We would like to know whether there is an effect of the type of music on ability to remember words. The most straight-forward way to find out is to estimate the parameters and then examine the posterior differences of the parameter estimates. The histograms in Figure 12.5 show the distributions of differences between the μ_j parameters. It can be seen that μ_1 is quite different than μ_3 and μ_4; a difference of zero falls well outside the 95% HDI intervals. From this we would conclude that Das Kruschke enhances memory better than Bach and Beethoven, but perhaps not better than Mozart.

A model-comparison approach addresses the issue in a different way. It compares the full model, which has distinct μ_j parameters for the four conditions, against a restricted model, which has a shared μ_0 parameter to describe all the conditions simultaneously. The two models have equal (50/50) prior probabilities. The upper panel of Figure 12.5 shows the results of a model comparison, using transdimensional MCMC with pseudopriors. The shared-μ_0 model is preferred, about 19 to 1 (0.951 to 0.049), over the distinct μ_j model. In other words, from this model comparison we would conclude that there is no difference in memory between the groups.

Which analysis should we believe? Is condition 1 different from conditions 3 and 4, or are all the conditions the same? Consider carefully what the model comparison actually says: Given the choice between one shared parameter and four different group parameters, the one-parameter model is less unbelievable. But that does not mean that the one-parameter model is the best possible model. In fact, if a different model comparison is conducted, that compares the one-parameter model against a different model that has one parameter for group 1 and a second parameter that is shared for groups 2, 3, and 4, then the comparison favors the two-parameter model (see for yourself in Exercise 12.1).

In principle, we could consider all possible models formed by partitioning the four groups. For 4 groups, there are 15 distinct partitions. We could, in principle, put a prior belief on each of the 15 models, and then do a comparison of the 15 models (Gopalan & Berry, 1998). From the posterior probabilities of the models, we could ascertain which partition was most believable, and decide whether it is more believable that other nearly-as-believable partitions. (Other approaches have been described by Berry & Hochberg, 1999; Mueller, Parmigiani, & Rice, 2007; Scott & Berger, 2006). Suppose that we conducted such a large-scale model comparison, and found that the most believable model partitioned groups 2–4 together, separate from group 1. Does this mean we should truly believe that there is zero difference between groups 2, 3, and 4? Not necessarily. If the groups are different, such as the four types of music in the present scenario, then there is almost certainly at least some small difference between their outcomes. (In fact, the simulated data do come from groups with all different means.) We may still want to estimate the magnitude of those small differences. An explicit posterior estimate will reveal the magnitude and uncertainty of those estimates. Thus, unless we have a reason to believe that different group parameters may be literally identical, an estimate of distinct group parameters will tell us what we want to know, without model comparison.

[1] To find information regarding Das Kruschke, search www.metalarchives.com. The author has no relation to the band, other than, presumably, some unknown common ancestor many generations in the past. The author was, however, in a garage band as a teenager. That band did not think it was playing death metal, although the music may have sounded that way to the critters fleeing the area.

12.3 Estimation or model comparison?

12.3.1 What is the probability that the null value is true?

There may be situations in which we want to know the probability that the null value is true. Proponents of the model-comparison approach may argue that the approach is especially applicable to these situations, because the model comparison yields the posterior probability of the null hypothesis. Unfortunately, the posterior probability of the null is only a relative one, depending completely on the alternative hypothesis that is used in the model comparison. For example, as mentioned earlier for the fictitious mind-reading experiment, a comparison of the null hypothesis (single-parameter) model against a four-parameter model yielded a strong preference in favor of the null hypothesis, but a comparison of the null hypothesis against a two-parameter model yielded a preference against the null hypothesis.

The posterior probability of the null hypothesis also depends crucially on the prior distribution assumed in the alternative model (e.g., Liu & Aitkin, 2008). Much of the effort in pursuing the model-comparison approach to null-hypothesis testing goes into justifying the form of the alternative-hypothesis prior (e.g., Edwards, Lindman, & Savage, 1963; Gallistel, 2009; Rouder, Speckman, Sun, Morey, & Iverson, 2009; Wagenmakers, 2007). Often the goal is to establish an "automatic" vague prior for the alternative model that has useful mathematical properties. The benefit of a conventionalized vague prior is that possible disputes regarding the best prior might be ignored by appealing to convention. Unfortunately, none of the automatic vague priors is usually the prior that is most appropriate for model comparison, which is an informed prior. A model comparison is useful if both models are genuinely viable, and viability is enhanced by incorporating prior information. For example, suppose a researcher is interested in investigating whether practitioners of therapeutic touch can detect the presence of another person's hand from a distance of a few centimeters (see Section 9.2.4, p. 177). The null hypothesis is that detection accuracy is at chance: $\theta = 0.5$. Should the alternative hypothesis, which entertains nonchance values, be set at an "automatic" uniform distribution? Proponents of therapeutic touch would probably say not; instead, the prior distribution should incorporate prior knowledge about how well the practitioners might do in this unusual task. The practitioners might argue that accuracy would typically be only a little above chance, perhaps at $\theta = 0.6$, with fair uncertainty, as expressed perhaps in a beta$(\theta|6,4)$ distribution.

The estimation approach, when combined with a ROPE, provides a different measure of the probability of the null. This probability is simply the proportion of the posterior distribution that falls within the ROPE. This proportion of the HDI within the ROPE is not the probability that the null value is true; instead, the proportion is the probability that the parameter is practically equivalent to the null value. Obviously, this probability depends enormously on the limits of the ROPE, and it is meaningful only if the limits of the ROPE are meaningful landmarks for the particular application. Also, the proportion of the HDI within the ROPE is most interpretable when it is large, because the proportion can be small for the trivial reason that the posterior is broad and shallow due to a small data set.

12.3.2 Recommendations

In general, Bayesian model comparison is only useful when both models are genuinely viable. If one or both models has little prior believability, then the Bayes factor and relative posterior believabilities are of little use. For example, suppose that we observe that there are

many nicely wrapped gifts strewn across the lawn beside the house. We might hypothesize two models for this observation: One hypothesis is that Santa Claus dropped them from the roof. Another hypothesis is that the Grinch dumped them from the roof. A Bayesian model comparison might prefer the Santa Claus hypothesis 50 to 1 over the Grinch hypothesis, but that does not mean we should believe in Santa Claus. Model comparison requires both models to have prior viability, not just one. To explain the gifts on the lawn, one hypothesis is that Santa Claus lost his grip (i.e., a non-viable hypothesis) and the other hypothesis is that a philosophically impressionable teenager in the household scattered the gifts in a fanatical fit of anti-materialist divestment (i.e., a viable but also unlikely hypothesis). A Bayesian model comparison might prefer the hypothesis of anti-materialist teenager, 50 to 1, but only by virtue of the fact that the alternative hypothesis was not viable in the first place. (Cf. Section 10.4.2.)

The premise that Bayesian model comparison is most meaningful when both models are genuinely viable implies that "automatic" Bayesian null-hypothesis testing might not be very meaningful, unless it is carefully applied only to situations in which both the null hypothesis is theoretically viable and the viable alternative hypothesis uses a reasonably informed prior. It is up to the user of the method to decide whether these conditions are met, and to carefully interpret the results of the comparison.

In general, Bayesian model comparison is a very useful tool, both inside and outside its specific application to null-hypothesis testing, as long as the models are viable, and as long as the models have priors that are equivalently informed. Even if both models are theoretically viable, if one model has a prior distribution on its parameters that is already close to the posterior, but the other model has a prior distribution on parameters that is far from the posterior, then the models are not starting with priors that are equivalently informed. These general points about Bayesian model comparison apply especially strongly to the specific case of Bayesian null-hypothesis testing.

There may be situations that specifically demand a dichotomous decision regarding a null value. In these situations, Bayesian null-hypothesis testing may be called for, but the alternative model and alternative prior should be carefully considered. Otherwise, the estimation approach can be both more informative and easier to implement.

12.4 R code

12.4.1 R code for Figure 12.5

This code is a straight-forward modification of the code from Section 10.2.2. Instead of the values being either distinct or shared across groups, the values are either distinct or shared across groups. In principle, the values could (or, indeed, should) simultaneously be either distinct or shared, but here the values are always distinct, merely to keep the code more readable. The only other difference from the code from Section 10.2.2 is that fictitious data are generated, whereby the groups have biases of .61, .50, .49, and .51, respectively.

(OneOddGroupModelComp.R)

```
#------------------------------------------------- -------------------------
# THE MODEL.

modelstring = "
# BUGS model specification begins here...
```

```
model {
    for ( i in 1:nSubj ) {
        # Likelihood:
        nCorrOfSubj[i] ~ dbin( theta[i] , nTrlOfSubj[i] )
        # Prior on theta (notice nested indexing):
        theta[i] ~ dbeta( aBeta[ CondOfSubj[i] ] , bBeta[ CondOfSub       j[i] ] )I(0.0001,0.9999)
    }
    # Re-parameterization of aBeta[j],bBeta[j] in terms of mu a        nd kappa:
    for ( j in 1:nCond ) {
        # Model 1: Distinct mu[j] each group.  Model 2: Shared mu0 all g      roups.
        aBeta[j] <-          ( mu[j]*(2-mdlIdx) + mu0*(mdlIdx-1) )    * kappa[j          ]
        bBeta[j] <- ( 1 - ( mu[j]*(2-mdlIdx) + mu0*(mdlIdx-1) ) ) * kap          pa[j]
    }
    # Hyperpriors for mu and kappa:
    for ( j in 1:nCond ) {
        mu[j] ~ dbeta( a[j,mdlIdx] , b[j,mdlIdx] )
    }
    for ( j in 1:nCond ) {
        kappa[j] ~ dgamma( shk , rak )
    }
    mu0 ~ dbeta( a0[mdlIdx] , b0[mdlIdx] )

    # Constants for hyperprior:
    # (There is no higher-level distribution of across-group re          lationships,
    # merely to keep the focus here on model comparison.)
    shk <- 1.0
    rak <- 0.1
    aP <- 1
    bP <- 1

    a0[1] <- .53*400       # pseudo
    b0[1] <- (1-.53)*400 # pseudo

    a0[2] <- aP # true
    b0[2] <- bP # true

    a[1,1] <- aP # true
    a[2,1] <- aP # true
    a[3,1] <- aP # true
    a[4,1] <- aP # true
    b[1,1] <- bP # true
    b[2,1] <- bP # true
    b[3,1] <- bP # true
    b[4,1] <- bP # true

    a[1,2] <- .61*100       # pseudo
    a[2,2] <- .50*100       # pseudo
    a[3,2] <- .49*100       # pseudo
    a[4,2] <- .51*100       # pseudo
    b[1,2] <- (1-.61)*100 # pseudo
    b[2,2] <- (1-.50)*100   # pseudo
    b[3,2] <- (1-.49)*100   # pseudo
    b[4,2] <- (1-.51)*100   # pseudo

    # Hyperprior on model index:
    mdlIdx ~ dcat( modelProb[] )
    modelProb[1] <- .5
    modelProb[2] <- .5
}
```

```
# ... end BUGS model specification
" # close quote for modelstring
# Write model to a file:
writeLines( text=modelstring , con="model.txt" )
# Load model file into BRugs and check its syntax:
modelCheck( "model.txt" )

#---------------------------------------------------                    ----------------------------
# THE DATA.

# For each subject, specify the condition s/he was in,
# the number of trials s/he experienced, and the number corre      ct.
# (Randomly generated fictitious data.)
npg = 20  # number of subjects per group
ntrl = 20 # number of trials per subject
CondOfSubj = c( rep(1,npg) , rep(2,npg) , rep(3,npg) , rep(4         ,npg) )
nTrlOfSubj = rep( ntrl , 4*npg )
set.seed(47401)
nCorrOfSubj = c( rbinom(npg,ntrl,.61) , rbinom(npg,ntrl,           .50) ,
                 rbinom(npg,ntrl,.49) , rbinom(npg,ntrl,.51) )
nSubj = length(CondOfSubj)
nCond = length(unique(CondOfSubj))
# Display mean number correct in each group:
for ( condIdx in 1:nCond ) {
      show( mean( nCorrOfSubj[ CondOfSubj==condIdx ] ) )
}

# Specify the data in a form that is compatible with BRugs mode     l, as a list:
datalist = list(
 nCond = nCond ,
 nSubj = nSubj ,
 CondOfSubj = CondOfSubj ,
 nTrlOfSubj = nTrlOfSubj ,
 nCorrOfSubj = nCorrOfSubj
)

# Get the data into BRugs:
modelData( bugsData( datalist ) )

#---------------------------------------------------                    ----------------------------
# INTIALIZE THE CHAINS.

nchain = 3
modelCompile( numChains=nchain )
modelGenInits()

#---------------------------------------------------                    ----------------------------
# RUN THE CHAINS.

burninSteps = 5000
modelUpdate( burninSteps )
samplesSet( c("mu","kappa","mu0","theta","mdlIdx") )
nPerChain = 5000 ; nThin = 10
modelUpdate( nPerChain , thin=nThin )
```

12.5 Exercises

Exercise 12.1. [Purpose: Model comparison for different partitions of group means.]

The program in Section 12.4.1 (OneOddGroupModelComp.R) does a model comparison in which M1 has different means for every group and M2 has the same mean for all groups. In this exercise, we consider a different model comparison, with a different partition of the conditions.

(A) Consider a comparison for which M1 has one mean for condition 1 and a second mean for conditions 2 through 4, and M2 has the same mean for all groups. A quick-and-dirty way to program this is by changing the model specification to (OneOddGroupModelCompEx12.1.R)

```
    for ( j in 1:2 ) {
      mu[j] ~ dbeta( a[j,mdlIdx] , b[j,mdlIdx] )
    }
    mu[3] <- mu[2]
    mu[4] <- mu[2]
```

Explain in English what the modified code does.

Unfortunately, the modified model structure is not amenable to automatic initialization in BUGS. We can "manually" initialize it this way: (OneOddGroupModelCompEx12.1.R)

```
genInitList <- function() {
  sqzData = .01+.98*datalist$nCorrOfSubj/datalist$nTrlOfSubj
  mu = aggregate( sqzData , list(datalist$CondOfSubj) , "mean" )[,"x"]
  sd = aggregate( sqzData , list(datalist$CondOfSubj) , "sd" )[,"x"]
  kappa = mu*(1-mu)/sd^2 - 1
  return(
    list(
      theta = sqzData ,
      mu = c( mu[1] , mean( mu[2:4] ) , NA , NA ) ,
      mu0 = mean(mu) ,
      kappa = kappa ,
      mdlIdx = 1
    )
  )
}
for ( chainIdx in 1 : nchain ) {
  modelInits( bugsInits( genInitList ) )
}
```

The initialization uses the same technique introduced in Section 9.5.2 (FilconBrugs.R), with only one novel nuance: The last two components of mu are specified as NA because those components are not stochastic, but are instead fixed (in the model specification) to equal mu[2].

Run the modified program, and show the resulting graphs analogous to those in Figure 12.5. Why are the upper histograms all the same, and why are the lower histograms (e.g., ω_2 vs ω_3) so strange looking?

(B) What should we conclude from the model comparison of the previous part? (Be careful to express your conclusion as a statement about relative believabilities.) Should we conclude that the means of conditions 2 through 4 are actually equal?

Exercise 12.2. [Purpose: Estimating a difference, including a ROPE (and the hot hand example with a better prior).]

(A) Consider again the hot-hand example from Exercise 8.1, p. 15this part of the exercise we establish a better prior: We know from generalskeball statistics that professional players tend to make about 75% of their free throwswith some players almost as low as 50% and some players almost as high as 95%. A beta(16,6) distribution nicely captures this prior knowledge. But what we do not know from statistics is the dierence between success after success and success after failurpairs of free throws. To be as vague as possible about thedierence, we'll put a uniform distribution on the difference. Notice that when the overall success rate is, say, 90%, thedierence could be anything from +20% (i.e., 100% vs. 80%) to20% (i.e., 80% vs. 100%), but when the overall success rate is, say, 70%, then the difference could be anything from+60% to 60%. The speci cation of the prior needs to accommodate this range that depends onoverall success. Here is one way to specify this in the BUGS model:

```
theta1 <- mu + deflect
theta2 <- mu - deflect
mu    dbeta( 16 , 6 )
delta      dbeta( 1 , 1 )
deflect <- (delta-.5)*2 * min(mu,1-mu)
```

The variablemuis the overall success rate. The variabledeflect is the de ection away from mucreated by a previous success or a previous failure. The valuedeflect is just a linearly transformed value of the random variabledelta , which has a uniform prior. Incorporate this code into the model and run without the data, so you can see the prior. Show your complete model speci cation and a graph of the MCMC sample, which should look much like the upper row of Figure 12.6.

(B) Suppose we establish a ROPE on thedierence of success-after-success and success-after-failure. We will arbitrarily set it at5%. Run the program with the data included, and display the posterior. TheplotPost.R function allows speci cation of a ROPE, with output as shown in Figure 12.6. Include your output graph and say in English what the ROPE indicates.

Exercise 12.3.[Purpose: Applying the approaches to a real-world example.] The thematic apperception test (TAT) is a method for assessing personality and other aspects of interpersonal attitudes. The subject is shown pictures of people in ambiguous scenes, and the subject is asked to make up a story about what the pictured people are doing. In a study reported by Werner, Stabenau, and Pollin (1970, Table 4),[2] pictures showing parent-child interactions were shown to women who had children. The stories invented by the mothers were scored for whether or not they expressed “personally involved, close-centered, exible interactions”. Twenty mothers of normal children and twenty mothers of schizophrenic children were each shown 10 pictures. The number of stories, out of 10, that expressed a exible interaction, were as follows.

Mothers of normal children: 8, 4, 6, 3, 1, 4, 4, 6, 4, 2, 2, 1, 3, 4, 3, 2, 6, 3, 4.
Mothers of schizophrenic children: 2, 1, 1, 3, 2, 7, 2, 1, 3, 0, 2, 4, 2, 3, 3, 0, 1, 2, 2.

For purposes of this exercise, we will assume that the prior is informed from previous research, which indicates that the typical number of stories that express exible interactions for this picture set is around 3 or 4, and can be well described by a beta(3.5,6.5) distribution. This is the distribution of exible-narrative tendencies across mothers.

(A) Estimate the dierence between TAT scores of the two groups of mothers. Be sur

[2]The author found these data by perusing the collection by Haley, Lunn, McConway, and Ostrowski (1994).

to use the prior knowledge. Is zero among the credible differences? If you assume that the difference has a ROPE from −0.1 to +0.1, is the ROPE excluded from the 95% HDI? HINTS: Set up a hierarchical model in BUGS, using the ltcat-condensation example as a guide. The prior applies to the distribution of individual theta[i] values, and the data are distributed as a binomial with bias theta[i].

(B) Using model comparison, determine the posterior probability of a model that uses one hyperparameter to describe both groups, relative to a model that uses a different hyperparameter for each group. For this application, the prior on the hyperparameters should not be informed, but instead should be extremely vague and broad, because the model comparison is supposed to be "automatic". What is the posterior probability of the different-hyperparameter model? If it is the preferred model, can we then use it to estimate the difference between groups? (Answer: Not necessarily, because it does not use the prior knowledge.)

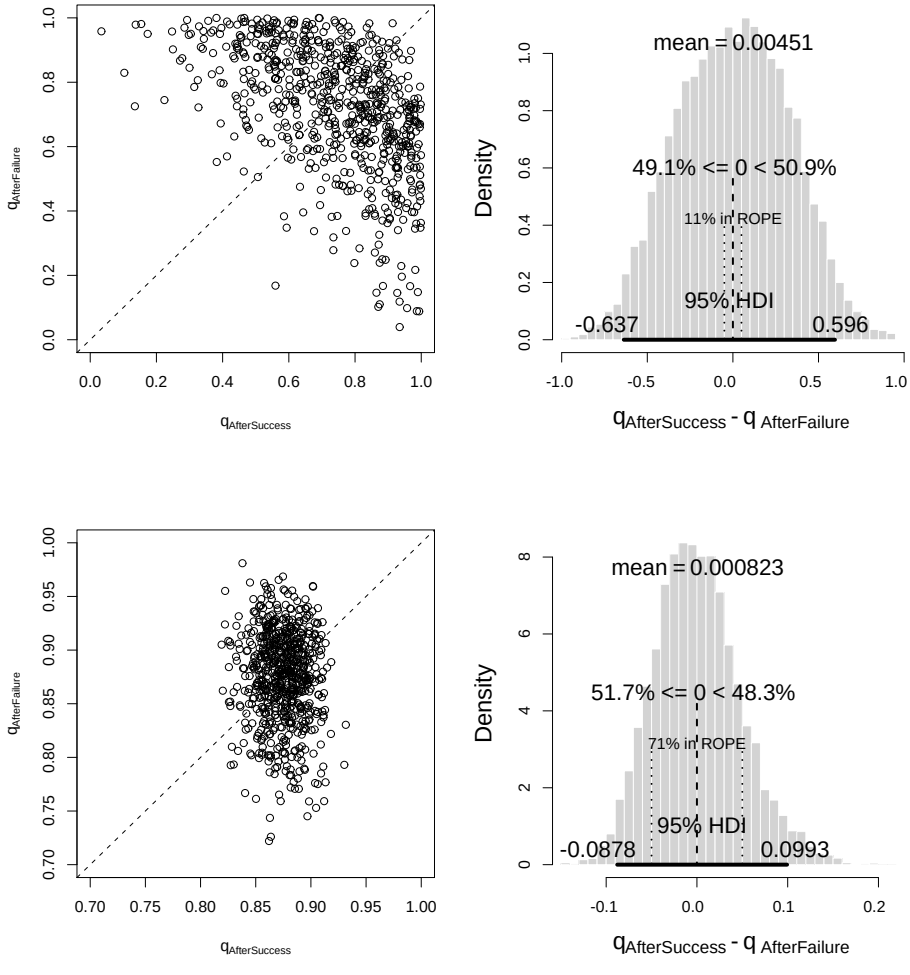

Figure 12.6: For Exercise 12.2. Upper row: The prior shows that the we believe the overall success rate is between 50% and 95%, but the prior difference between the two 's could be anything. Lower row: The posterior. Notice the ROPE on the difference, from :05 to +:05, along with the display of what percentage of the posterior falls within the ROPE.

Chapter 13

Goals, Power, and Sample Size

Contents

> Just how many times must I show her I care,
> Until she believes that I'll always be there?
> Well, while she denies that my value's enough,
> I'll have to rely on the power of love[1].

Researchers collect data in order to achieve a goal. Sometimes the goal is to show that a suspected underlying state of the world is believable; other times the goal is to put a minimal degree of precision on whatever trends are observed. The goal can only be probabilistically achieved, as opposed to definitely achieved, because data are incomplete with random noise that can obscure the underlying state of the world. Statistical power is the probability of achieving the goal of an empirical study, when a suspected underlying state of the world is true. Scientists don't want to waste time and resources pursuing goals that have a small probability of being achieved. In other words, scientists should consider power in their experiments.

[1] The power of "love"? Sailors know that there's not much power in love.

13.1 The Will to Power

In this section, a framework for research and data analysis is described, which leads to a more precise de nition of power and how to compute it.[2]

13.1.1 Goals and Obstacles

There are many possible goals for an experiment or observational study. For example, we might want to show that the rate of recovery for patients who take a drug is higher than the rate of recovery for patients who take a placebo. We might want to show that a coin is biased, i.e., that its tendency to show heads is not equal to 50%. Goals such as those involve showing that a suspected difference or value really is tenable, or, complementarily, showing that a "null" value is not tenable. Other goals do not necessarily have a suspected value in mind. Instead, the goal is merely to put a desired degree of precision on whatever tendencies happen to be observed. For example, when polling a population of voters' preferences of political candidates, the goal is not necessarily to show that any particular candidate is ahead, but to get a precise assessment of the populations's preferences.

The various goals of research can be formally expressed in various ways. In this chapter I will focus on two goals, formalized in terms of the HDI.

Goal: Demonstrate that believable parameter values exclude a "null" value.

- Formal expression: Show that the 95% HDI excludes the "null" value, or its ROPE.

Goal: Achieve precision in the estimate of believable values.

- Formal expression: Show that the 95% HDI has some specified maximal width.

In some research, the goal may be to demonstrate that a "null" value is true, rather than false. This goal can also be formalized in terms of precision of the HDI: We want the entire 95% HDI to fall within the ROPE, as described in Section 12.1 (p. 244). There are other mathematical formalizations of the various goals, and they will be mentioned later. This chapter focuses on the HDI because of its natural interpretation for purposes of parameter estimation, as explained in the previous chapter.

If we knew the benefits of achieving our goal, and the costs of pursuing it, and if we knew the penalties for making a mistake while interpreting the data, then we could express the results of the research in terms of the long-run expected payoff. When we know the costs and benefits, we can conduct a full decision-theoretic treatment of the situation, and plan the research and data interpretation accordingly (e.g., Chaloner & Verdinelli, 1995; Lindley, 1997). In our applications we do not have access to those costs and benefits, unfortunately. Therefore we rely on goals such as those outlined above.

The crucial obstacle to the goals of research is that a random sample is only a probabilistic representation of the population from which it came. Even if a coin is actually fair, a random sample of flips will rarely show exactly 50% heads. And even if a coin is biased, it might come up heads 5 times in 10 flips. Drugs that actually work better than a placebo might happen to cure more patients in a particular random sample. And drugs that truly are

[2] Regarding the title of this section: Other than the fact that researchers desire statistical power, the notion of statistical power might have profound connections with concepts from Friedrich Nietzsche's work, The Will to Power. See Exercise 13.1.

e ective might happen to show little dierence from a placebo in another particular random sample of patients. Thus, a random sample is a ckle indicator of the true value in the underlying world. Whether the goal is showing that a suspected value is or isn't believable, or showing that a precise range of values is tenable, random variation is the researcher's bane. Noise is the nemesis.

13.1.2 Power

Because of random noise, the goal of a study can be achieved only probabilistically. The probability of achieving the goal, given the (suspected) true state of the world, is called the power of the experiment.

Scientists go to great lengths to try to increase the power of their experiments or observational studies. There are three primary methods by which researchers can increase the chances of detecting an ect. First, we reduce measurement noise as much as possible. For example, if we are trying to determine the cure rate of a drug, we try to reduce other random inuences on the patients, such as other drugs they might be taking, changes in diet or rest, etc. Reduction of noise and control of other inuences is the primary reason for conducting experiments in the lab instead of in the messy contexts of the real world. The second method, by which we can increase the chance of detecting an ect, is to amplify the underlying magnitude of the ect if we possibly can. For example, if we are trying to show that a drug helps cure a disease, we will want to administer as large a dose as possible (assuming there are no negative side ects). In non-experimental research, in which the researcher does not have the luxury of manipulating objects being studied, this second method is unfortunately unavailable. Sociologists, economists, and astronomers, for example, are often restricted to observing events that they cannot control or manipulate.

Once we have done everything we can to reduce noise in our measurements, and to amplify the e ect we are trying to measure, the third way to increase power is to increase the sample size. The intuition behind this method is simple: With more and more measurements, random noises will tend to cancel themselves out, leaving on average a clear signature of the underlying ect. In general, as sample size increases, power increases. Increasing the sample size is an option in most experimental research, and in a lot of observational research (e.g., more survey respondents can be polled), but not in some domains where the population is nite, such as comparative studies of the states or provinces of a nation. In this latter situation, we cannot create a larger sample size, but Bayesian inference is still valid, and perhaps uniquely so (Western & Jackman, 1994).

What we accomplish in this chapter is precise calculation of power. We compute the probability of achieving a speci c goal, given (i) a suspected underlying distribution of biases or ects in the population being measured, and (ii) a speci ed data-generating process, such as collecting a xed number of observations. Power calculations are very useful for planning an experiment. To plan our research, we conduct "dress rehearsals" before the actual performance. We repeatedly simulate data we suspect we might get, and conduct Bayesian analyses on the simulated data sets. If the goal is achieved for most of the simulated data sets, then the planned experiment has high power. If the goal is rarely achieved in the analyses of simulated data, then the planned experiment is likely to fail, and we must do something to increase its power.

In general, power can be approximated in the following manner:

1. Generate a random sample of data points, using the data-generating hypothesis. The sample should be generated according to how actual data would be gathered in the

eventual real experiment. For example, typically it is assumed that the number of data points is fixed at N. It might instead be assumed that data will be collected for a fixed interval of time T, during which data points appear randomly at a known mean rate n/T.

2. Compute the posterior estimate, using Bayesian analysis with skeptical-audience priors. The data analysis must be convincing to the audience, and therefore the analysis must use priors that are agreeable to that skeptical audience.

3. From the posterior estimate, tally whether or not the goal was attained. The goal could be any of those outlined previously, such as having 95% HDI exclude a ROPE around the null value, or having the 95% HDI be narrower than a desired width.

4. Repeat above steps many times, to approximate the power. Power is, by definition, the proportion of times that the goal is attained.

The result of this process is the power, i.e., the probability of achieving the goal when using the particular data-sampling procedure. Notice that if the data-sampling procedure uses a fixed sample of size N, then the process determines power as a function of N. If the data-sampling procedure uses a fixed sampling duration T, then the process determines power as a function of T.

Figure 13.1 illustrates the process of computing power. The upper part of the figure shows the flow of information in an actual analysis. The real world generates a single sample of observed data. We use Bayes' rule, starting with a prior for the skeptical audience, to derive an actual posterior distribution. The lower part of the figure shows the flow of information in a power analysis. Instead of the real world, we have a hypothesis about the world. The hypothesis is expressed as a belief distribution over the parameters of the model. In many cases, this hypothesis is a posterior distribution derived from previous research. From the hypothesis, a sample of data is generated. This simulated sample of data is exemplary of data we anticipate if the the hypothesis is true. We then conduct a Bayesian analysis on the anticipated data sample, thereby deriving an anticipated posterior distribution. For the anticipated posterior, we assess whether or not the goal was achieved. We then repeat the simulated data generation process. The iteration is indicated in Figure 13.1 by the layers of anticipated data samples and anticipated posteriors. Across repetitions, the proportion of times that the goal is achieved is the power. For some simple situations, power can determined exactly, as is done, for example, in the program of Section 13.6.1 (minNforHDIpower.R). In more realistic situations, power is approximated through simulation (Wang & Gelfand, 2002), as is done, for example in the program of Section 13.6.2 (FilconBrugsPower.R).

13.1.3 Sample Size

Power increases as sample size increases (usually). Because gathering data is costly, we would like to know the minimal sample size, or sample duration, that is required to achieve a desired power.

The goal of precision in estimation can always be attained with a large enough sample size. This is because the likelihood of the data, thought of graphically as a function of the parameter, tends to get narrower and narrower as sample size increases. This narrowing

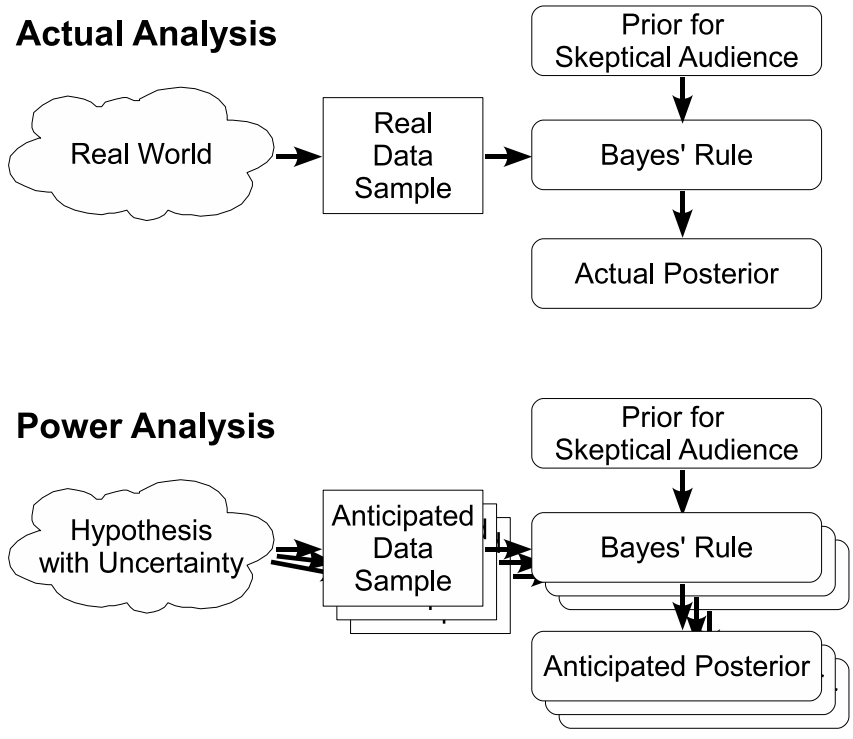

Figure 13.1: Upper diagram illustrates the flow of information in an actual Bayesian analysis. Lower diagram illustrates flow of information in a power analysis.

of the likelihood is also what causes the data to eventually overwhelm the prior beliefs as sample size increases. As we collect more and more data, the likelihood function gets narrower and narrower (on average), and therefore the posterior gets narrower and narrower. Thus, with a large enough sample, we can make the posterior distribution as narrow, i.e., precise, as we like.

The goal of showing that a parameter value is credibly different from a null value might not be attainable with a high enough probability, however, no matter how big the sample size. Whether or not this goal is attainable with high probability depends on the data-generating distribution. The best that a large sample can do is exactly reflect the data-generating distribution. If the data-generating distribution has considerable mass straddling the null value, then the best we can do is get estimates that include or straddle the null value. As a simple example, suppose that we think that a coin may be biased, and the data-generating hypothesis is that $p(\theta = .5) = 25\%$, $p(\theta = .6) = 25\%$, $p(\theta = .7) = 25\%$, and $p(\theta = .8) = 25\%$. Because 25% of the simulated data come from a fair coin, the maximal probability of excluding $\theta = .5$, even with a huge sample, is 75%.

Therefore, when planning the sample size for an experiment, it is crucial to first decide what a realistic goal is. If there are good reasons to posit a highly certain data-generating hypothesis, perhaps because of extensive previous results, then a viable goal may be to exclude a null value. On the other hand, if the data-generating hypothesis is somewhat vague, then a more reasonable goal is to attain a desired degree of precision in the posterior.

13.1.4 Other Expressions of Goals

There are other ways to express mathematically the goal of precision in estimation. For example, another way of using HDIs was described by Joseph, Wolfson, and du Berger (1995a, 1995b). They considered an "average length criterion" which requires that the average HDI width, across repeated simulated data, does not exceed some maximal value L. There is no explicit mention of power, i.e., the probability of achieving the goal, because the sample size is chosen so that the goal is definitely achieved. The goal itself is probabilistic, however, because it regards an average: While some data sets will have HDI width less than L, many other data sets will not have an HDI width less than L. Another goal considered by Joseph et al. (1995a) was the "average coverage criterion." This goal starts with a specified width for the HDI, and requires its mass to exceed 95% (say) on average across simulated data. The sample size is chosen to be large enough to achieve the goal. Again, power is not explicitly mentioned, but the goal is probabilistic. Some data sets will have an HDI width HDI mass greater than 95%, and other data sets will have an HDI width HDI mass greater than 95%. Other goals regarding precision are provided by Adcock (1997) and by De Santis (2004, 2007). The methods emphasized in this chapter focus on limiting the worst precision, instead of the average precision.

A rather different mathematical expression of precision is entropy of a distribution. Entropy describes how spread out a distribution is, so smaller entropy connotes a more precise distribution. A distribution that consists of an infinitely dense spike, that has an infinitesimally narrow width, has zero entropy. At the opposite extreme, a uniform distribution has maximal entropy. The goal of high precision in the posterior distribution might be re-expressed as a goal of small entropy in the posterior distribution. For an overview of this approach, see, e.g., Chaloner and Verdinelli (1995). For an introduction to how minimization of expected entropy might be used spontaneously by people as they experiment with the world, see Kruschke (2008). Entropy may be a better measure of posterior precision than HDI width especially in cases of multimodal distributions, for which HDI width is more challenging to determine. I will not further explicate the use of entropy because I think that HDI width is a more intuitive quantity than entropy by which to express precision, at least for most researchers in most contexts.

There are also other ways to express mathematically the goal of excluding a null value. In particular, the goal could be expressed as wanting a Bayes factor in a model comparison between the spike-null prior and the automatic alternative prior (e.g., Wang & Gelfand, 2002; Weiss, 1997). For example, we might set the desired Bayes factor at a ratio of 19 to 1 (i.e., .95 to .05). Kass and Raftery (1995) provide some guidelines for choice of criterial Bayes factor. I will not further address this approach, however, because the goal of a criterial BF for untenable caricatured priors has problems as discussed in the previous chapter. Instead, it will be assumed that the goal of the research is estimation of the parameter values, starting with a viable prior. The resulting posterior is then used to assess whether the goal was achieved.

13.2 Sample size for a single coin

As our first worked-out example, consider the simplest case: Data from a single coin. Perhaps we are polling a population and we want to know if there is preference for candidate A or candidate B. Or perhaps we want to know if a drug has more than 50% cure rate. We will go through the steps listed in Section 13.1.2 to compute the exact sample size needed

Table 13.1: Minimal sample size required for 95% HDI to exclu 0.5, when ipping a single coin.

	Generating Mean						
Power	.55	.60	.65	.70	.75	.80	.85
.7	642	148	64	33	18	13	7
.8	866	191	82	43	26	18	10
.9	1274	264	111	59	36	24	16

Note. The data-generating distribution is a beta density wi mean , as indicated by the column header, and with shape parameter of and (1) , where = 2000. The audience-agreeable prior is uniform.

to achieve various degrees of power for erent data-generating hypotheses.

13.2.1 When the goal is to exclude a null value

Suppose that our goal is to show that the coin is biased. In words, we want to show that .5 is not among the credible values. More speci cally, we want to show that the 95% HDI excludes = :50.

Because of previous research (or just a really strong hunch), we think that the true bias of the coin is very close to = :65. It is a strong belief, so we'll describe the data-generating hypothesis as a beta distribution with mean .65 and certainty based on 2,000 ctitious ips of the coin. This puts 95% of the data-generating biases between .63 and .67.

Next, we generate simulated data from hypothetical coins that have those biases, and tally how often the HDI excludes = :50. One replication of the process goes like this: First, select a value for the "true" bias in the coin, from the data-generating distribution that is narrowly centered on = :65. Suppose that the selected value is .638. Second, simulat ipping a coin with that bias N times. The simulated data have z heads and N z tails. The proportion of heads, z=N, will tend to be around .638, but will be higher or lower because of randomness in the ips. Third, using the audience-agreeable prior for purposes of data analysis, determine the posterior beliefs regarding if z heads in N ips were observed. Tally whether or not the 95% HDI excludes the null value of = :50. Notice that even though the data were generated by a coin with bias of .638, the data might, by chance, show a low proportion of heads, and therefore the 95% HDI might exclude = :50. This process is repeated many times to estimate the power of the event, i.e., the probability that the null value will be excluded from the HDI. In fact, for this simple situation, the exact power can be computed analytically, as explained in Section 13.2.1 (minNforHDIpower.R), p. 272.

Table 13.1 shows the minimal sample size needed for the 95% HDI to exclude = :5 when ipping a single coin. As an example of how to read the table, suppose you have a data-generating hypothesis that the coin has a bias very near = :65. This hypothesis is implemented, for purposes of Table 13.1, as a beta distribution with shape parameters of :65 2000 and (1 :65) 2000. The value of 2000 is arbitrary; it's as if the generating mean of .65 was based on ctitious previous data containing 2000 ips. The table indicates that if we desire a 90% probability of obtaining a 95% HDI that excludes = :5, we need a sample size of N = 111, i.e., we need to ip the coin at least 111 times in order to have a 90% chance that = :5 falls outside the 95% HDI.

Notice, in Table 13.1, that as the generating mean increases, the required sample size decreases. This makes sense intuitively: When the generating mean is large, the sample proportion of heads will tend to be large, and so the HDI will tend to fall toward the high end of the parameter domain. In other words, when the generating mean is large, it doesn't take a lot of data for the HDI to fall well above $= :5$. On the other hand, when the generating mean is only slightly above $= :5$, then it takes a large sample for the sample proportion of heads to be consistently above .5, and for the HDI to be consistently entirely above .5.

Notice also, in Table 13.1, that as the desired power increases, the required sample size increases quite dramatically. For example, if the data-generating mean is .6, then as the desired power rises from .7 to .9, the minimal sample size rises from 148 to 266.

13.2.2 When the goal is precision

Not all research has as its goal the exclusion of a particular value. Sometimes the goal is to establish a precise estimate of the parameter value. Sometimes, the research may have a null value of interest, but the data-generating hypothesis is too vague for the null value to be excluded most of the time, or, the goal may be to demonstrate that the HDI falls entirely within the ROPE. In these cases, the goal becomes precision of parameter estimation.

Here is an example in which there is a null value of interest, but only a vague data-generating distribution. Suppose you are interested in assessing the preferences of the general population regarding political candidates A and B. In particular, you would like to have high con dence in estimating whether the preference for candidate A exceeds $= :5$. A recently conducted poll by a reputable organization found that of 10 randomly selected voters, 6 preferred candidate A and 4 preferred candidate B. If we use a uniform pre-poll prior, our post-poll estimate of the population bias is a $(7; 5)$ distribution. As this is our best information about the population so far, we can use the beta(7,5) distribution as a data-generating distribution for planning the follow-up poll. Unfortunately, a beta(7,5) distribution has a 95% HDI from $= :318$ to $= :841$, which means that $= :5$ is well within the data-generating distribution. How many more people do we need to poll so that 80% of the time we would get a 95% HDI that falls above $= :5$?

It turns out, in this case, that we can never have a sample large enough to achieve the goal of 80% of the HDIs falling above $= :5$. To see why, consider what happens when we sample a particular value from the data-generating distribution, e.g., $= :4$. We use that value to simulate a random sample of votes. Suppose the sample is huge, which implies that the HDI will be very narrow. What value of will the HDI focus on? Almost certainly it will focus on the value $= :4$ that was used to generate the data. To reiterate, when is very large, the HDI essentially just reproduces the value that generated it. Now recall the data-generating distribution of our example: The $(; 7; 5)$ has only about 72% of the values are above .5. Therefore, even with an extremely large sample size, we can get at most 72% of the HDIs to fall above .5.

There is a viable alternative goal, however. Instead of trying to reject a particular value of , we set as our goal a desired degree of precision in the posterior estimate. For example, our goal might be that the 95% HDI has width less than 0.2, at least 80% of the time. This goal implies that regardless of what values of happen to be emphasized by the posterior distribution, the width of the posterior is usually narrow, so that we have attained a suitably high precision in the estimate.

Table 13.2 shows the minimal sample size needed for the 95% HDI to have maximal

Table 13.2: Minimal sample size required for 95% HDI to have maximal width of 0.2, when ipping a single coin.

	Generating Mean						
Power	.55	.60	.65	.70	.75	.80	.85
.7	91	91	89	86	80	69	58
.8	92	92	91	90	86	79	67
.9	93	93	93	92	91	88	79

Note. The data-generating distribution is a beta density with mean , as indicated by the column header, and with shape parameter of and (1) , where = 10. The audience-agreeable prior is uniform.

width of 0.2. As an example of how to read the table, suppose you have a data-generating hypothesis that the coin has a bias roughly around .6. This hypothesis is implemented, for purposes of Table 13.2, as a beta distribution with shape parameters of 6 10 and (1 :6) 10. The value of 10 is arbitrary; it's as if the generating mean of .6 were based on ctitious previous data containing only 10 ips. The table indicates that if we desire a 90% probability of obtaining an HDI with maximal width of .2, we need a sample size of at least 93.

Notice in Table 13.2 that as the desired power increases, the required sample size increases only slightly. For example, if the data-generating mean is .6, then as the desired power rises from .7 to .9, the minimal sample size rises from 91 to 93. This is because the distribution of HDI widths, for a given sample size, has a steeply shunted high tail, and therefore small changes in can quickly pull the high tail across a threshold such as .2. On the other hand, as the desired HDI width decreases (not shown in the Table), the required sample size increases rapidly. For example, if the data-generating mean is .6 with = 10, and the desired HDI width is 0.1 (instead of 0.2), then the sample size needed for 80% power is 377 (instead of 92).

The R code in Section 13.6.1 (NforHDIpower.R), generated Tables 13.1 and 13.2. You can use the R code to specify your own data-generating distribution, goal, and desired power. You can also specify a ROPE. The R function will tell you the minimal sample size required. Exercise 13.2 has you work through some examples.

13.3 Sample size for multiple mints

Consider again the ltration-condensation experiment described in Section 9.3.1, p. 178, in which people learned four different category structures. Two structures were " ltration" structures in which one stimulus dimension could be ignored but perfect accuracy could be achieved. Two other structures were "condensation" structures in which both stimulus dimensions required processing to achieve perfect accuracy. Some theories of learning predict that ltration should be easier to learn than condensation, and therefore one goal of the experiment was to show that the mean accuracy of people in the ltration structure di ered from the mean accuracy of people in the condensation structure. This goal was handily achieved: Using = 40 participants per condition, the 95% HDI on the difference of parameters went from 0.173 to 0.259, with 100% of the MCMC differences falling well above 0.0; see the right-most panel of Figure 9.16, p. 18

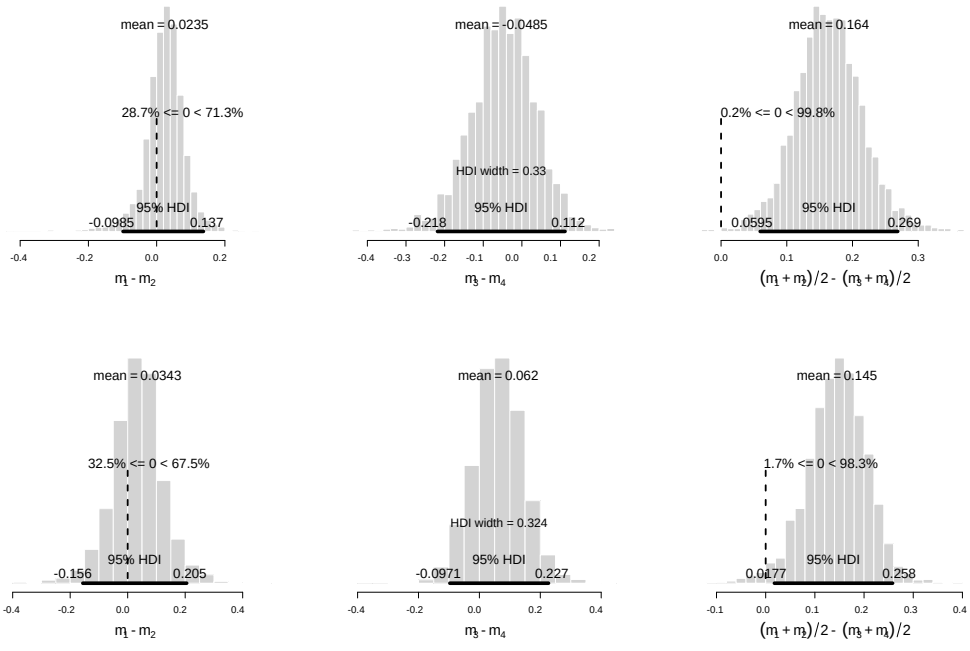

Figure 13.2: Posterior distributions from two runs of simulated data, using $N = 6$ per group. Upper row (right panel) shows a case in which the posterior for $(\mu_1 + \mu_2)/2 - (\mu_3 + \mu_4)/2$ excludes 0.0, as desired. Lower row (right panel) shows a case in which that goal is not achieved. Across many simulations, over 80% achieved the goal.

Is there a smaller sample size per group that could have been used to achieve high power? And what was the power of detecting the more subtle difference between the two Ltration groups, shown in the left-most panel of Figure 9.1, p. 181? To answer these questions, we pursue the simulation procedure outlined at the beginning of this chapter (Section 13.1.2): Specify a data-generating hypothesis, simulate some data, do a Bayesian data analysis, determine whether the goal is achieved, and repeat many times to estimate the probability of achieving the goal. R code for computing power for this experiment is listed in Section 13.6.2 (IconBrugsPower.R).

Suppose the goal of the experimenter is to show that the mean of the Ltration groups exceeds the mean of the condensation groups. Specifically, suppose that the goal is achieved if the 95% HDI of $(\mu_1 + \mu_2)/2 - (\mu_3 + \mu_4)/2$ excludes 0.0. It turns out that using merely $N = 6$ in each of the groups achieves over 80% power for this goal. Figure 13.2 shows examples of the posteriors for two runs of simulated data with $N = 6$ per group. The upper row shows a case in which the posterior for $(\mu_1 + \mu_2)/2 - (\mu_3 + \mu_4)/2$ excludes 0.0, as desired. The lower row shows a case in which that goal was not achieved. Across many simulated runs using $N = 6$, 80% achieved the goal. In other words, instead of using $N = 40$, as in the actual experiment, the research could have been conducted with only $N = 6$, and had an 80% chance of inferring a believable difference between Ltration and condensation.

This power analysis stems from simulated data that have been generated from moderately certain knowledge, based on results from 40 actual subjects per condition. If we did

Table 13.3: Types of power analysis, including replication probability.

Analysis Type	Data Generator	Data Sample	Prior for Bayesian Analysis	Posterior
Actual	Real World	Observed Once	Skeptical Audience	Actual
Prospective Power	Hypothesis	Simulated Repeatedly	Skeptical Audience	Anticipated
Retrospective Power	Actual Posterior	Simulated Repeatedly	Skeptical Audience	Anticipated
Replication Power	Actual Posterior	Simulated Repeatedly	Actual Posterior	Anticipated

not have data-generating priors with such high certainty, then our power predictions would not be so high either. For example, if we had data-generating priors on β_j with the same means as used here, but with wider (more uncertain) distributions, then the sample size would have to be much bigger than $N = 6$ to achieve the same power. This makes intuitive sense: If we are not very certain about what the data will look like, it takes a lot of data to reliably reveal the underlying trends.

Suppose that the goal of the experimenter was, instead, to show that there was a difference between the two habituation groups. Specifically, we suppose that the goal is achieved if the 95% HDI of $\beta_1 - \beta_2$ excludes 0.0. Notice that this goal was not achieved in either simulated run shown in Figure 13.2. It turns out that using $N = 6$ in each of the groups achieves a power of only 4% for this goal. What is the power for this goal if $N = 40$, as used in the actual experiment? It turns out that the power is only 43%! The fact that the actual data (Figure 9.16, p. 181) achieved the goal was merely a lucky happenstance.

Finally, suppose that the goal of the experimenter was to achieve a minimal precision on the difference between the two condensation groups. The motivation for this goal is as follows: Theory suggests that the two condensations groups should not differ much. Therefore the goal cannot be to show that the difference exceeds zero. Instead, the goal should be to achieve some desired degree of precision regarding the difference. Suppose we want the 95% HDI on $\beta_3 - \beta_4$ to have a width less than 0.20. To achieve this goal at least 80% of the time, i.e., with a power of 80%, we would need a sample size (per group) of approximately $N = 32$. You can verify this by running the program in Section 13.3.6 (FilconBrugsPower.R).

13.4 Power: prospective, retrospective, and replication

There are different types of power analysis, as shown in Table 13.3. For reference, the top row of the table shows an actual analysis, in which the data generator is the real world, the data sample is observed once, the prior for the analysis is intended to be agreeable to a skeptical audience, and the actual posterior distribution is computed.

The second row of Table 13.3 shows prospective power analysis, for which the data generator is a research hypothesis based on theory and/or results from previous experiments that are not exactly the same as the present experiment. The research hypothesis is expressed as a distribution over model parameters, with a tendency and uncertainty

that captures the researcher's beliefs about what would be appropriate parameter values if the data of the planned experiment were to appear as it is believed they would. The distribution over the model parameters could be set directly, from theoretical considerations, by the researcher setting the shape constants of the parameter distribution. On the other hand, the data-generating distribution could be set indirectly, by first positing data consistent with the theory and then applying Bayesian updating to derive a distribution over parameters that is consistent with the posited data. De Santis (2007) describes how to combine results of previous experiments to construct a data-generating distribution.

Contrast a prospective power analysis with the next row of the table, which shows a retrospective power analysis. This refers to a situation in which we have already collected data, and therefore we already have an actual posterior distribution. We want to determine the power of the experiment we ran. To do this, we use the actual posterior as the data generator. This is tantamount to using the data from a posterior predictive check. In other words, at a step in the chain, the parameter values are used to generate simulated data. The simulated data are then analyzed with the same Bayesian model as the actual data, and the posterior is examined for whether or not the goal is achieved. This process is readily programmed in BRugs, as explained in Section 13.6.2 (FilconBrugsPower.R). The example of the previous section, regarding the filtration-condensation experiment, was a case of retrospective power analysis.

Finally, suppose that we have already collected some data and we have an actual posterior distribution, and we want to know the probability that we would achieve our goal if we replicated the experiment. In other words, if we were to simply re-collect a new sample of data, what is the probability that we would achieve our goal in the replicated study, also taking into account the first set of data? The bottom row of Table 13.3 indicates how we answer this question of replication power. We use the actual posterior derived from the first sample of data as the data generator. But for the analysis prior, we again use the actual posterior from first sample of data, because that is the best-informed prior for the follow-up experiment. An easy way to execute this analysis in BUGS is as follows: Use the actual set of data with a skeptical-audience prior to generate a sample from an actual posterior. Use that actual-posterior sample to generate simulated new data, as if in a posterior predictive check. Then, concatenate the original data with the novel simulated data and update the original skeptical-audience prior with the merged data set. This technique is tantamount to using the posterior of the original data set as the prior for the novel simulated data. Section 13.6.2 (FilconBrugsPower.R) shows BRugs code for this type of analysis.

Computation of replication power is natural in a Bayesian setting, but is difficult or impossible for traditional null-hypothesis significance testing (Miller, 2009). NHST has trouble when addressing replication probability because it has no good way to model a data generator: It has no access to the posterior distribution from the initial analysis.

13.4.1 Power analysis requires verisimilitude of simulated data

Power analysis is only useful when the simulated data imitate actual data. We generate simulated data from a descriptive model that has uncertainty in its parameter values, but we assume that the model is a reasonably good description of the actual data. If the model is instead a poor description of the actual data, then the simulated data do not imitate actual data, and inferences from the simulated data are not very meaningful. It is advisable, therefore, to perform a posterior predictive check that the simulated data accurately reflect the actual data.

When simulated data differ from actual data, strange results can arise in power analysis. Consider an analysis of replication probability in which the simulated data are quite different than the actual data. The novel simulated data are combined with the original data to conduct the replication analysis. The combined data set is a mixture of two different trends (i.e., the actual trend and the different simulated trend), and therefore the estimates of the parameters become more uncertain than for the original data alone. It is only when the simulated sample size becomes large, relative to the original sample size, that the simulated trend overwhelms the actual trend, and the replication uncertainty becomes smaller again. If you find in your power analyses that parameter uncertainty initially gets larger as the simulated sample size increases, then you may have a situation in which the model does not faithfully mimic the actual data.

13.5 The importance of planning

In the real conduct of research, scientists do not always use a specific sample size and then stick to it. Nor should they have to. When experiments are used such that the results from one participant do not affect the next participant, or when observational studies are made such that the results of one observation are independent of other observation —in other words when the next coin flip is not influenced by previous flips— then the researcher's intentions do not influence the data production or the data interpretation.

In some situations, data are collected for a particular period, during which the actual number of participants (coin flips) is a random value that depends on the random rate at which people happen to participate. For example, participants in a typical university psychology experiment might be recruited for a certain number of weeks during a semester. There may be a stable average rate of participation, but the number of participants in any given week is a random number. In this situation, our goal is to determine how many weeks we should plan on recruiting participants, so to have a high probability of achieving the goal of the experiment. We can do the power analysis in a manner analogous to the methods of the previous sections. As in those methods, we randomly generate parameter values from our data-generating hypothesis, but the sample size is also a random value that depends on a specified rate of participation and amount of time during which participants are recruited. The sample size could be (but is not necessarily) modeled as a Poisson variable, as was done in Exercise 11.3. We simulate incrementally longer durations until we find the minimal duration that yields a sufficiently high power. Having thereby planned a duration for collecting data (and corresponding approximate number of subjects), we can decide whether or not it is feasible to actually conduct the research. If we do go ahead with the research, we are not stuck with the exact planned duration. We can stop whenever we want, assuming that the data are uninfluenced by our intentions to stop.

The point of the previous paragraphs is to say that sticking to a planned sample size or duration is not crucial to the data analysis, after the data have been collected. Instead, the planning is important for deciding whether or not the research has a reasonable chance of success, before the data are collected.

Conducting a power analysis in advance of collecting data is very important and valuable. Often in real research, a fascinating theory and clever experimental manipulation imply a subtle effect. It can come as a shock to the researcher when power analysis reveals that detecting the subtle effect would take several hundred subjects! But the shock of power analysis is far less than the pain of running dozens of subjects and finding highly uncertain

estimates of the sought-after effect.

Power analysis can reduce research pain in other ways. Sometimes in real research, an experiment or observational study is conducted merely to subjectively confirm what is anecdotally known to be a strong effect. A researcher may be tempted to conduct a study using the usual large sample size that is typical of related research. But a power analysis may reveal that the strong effect can be easily detected with a much smaller sample size.

Power analysis is also important when proposing research to funding agencies. Proposals in basic research might have fascinating theories driving the predicted effects are subtle, then reviewers of the proposal may be justifiably want to be reassured by a power analysis. Proposals in applied research are even more reliant on power analysis, because the costs and benefits are immediate and tangible. For example, in clinical research (e.g., medicine, pharmacology, psychiatry, counseling), it can be very costly to test patients, and therefore it is important to anticipate the probable sample size or duration.

While it is important to plan sample size in advance, it can also be important, especially in clinical applications, to monitor data as they are collected and to stop the research as soon as possible. It behooves the researcher to discontinue an experiment as soon as the data clearly indicate a positive or negative outcome: It would be unethical to slavishly continue treating patients with an experimental treatment that is clearly detrimental, and it would be unethical to slavishly continue running patients in a placebo condition when the experimental treatment is clearly having positive effects. If the data are collected in such a way that they are uninfluenced by the experimenter's intentions, then the data collection can be stopped at any time without influencing the interpretation of the data. The decision regarding when to stop collecting data is a topic of much subtlety and sophistication, and goes under the name of Bayesian sequential design. It will not be discussed further here, but the interested reader is referred to books by J. O. Berger (1985) and DeGroot (2004), and various technical articles, for example those cited by Royal, Ghosal, and Rosenberger (2009, p. 427).

13.6 R code

13.6.1 Sample size for a single coin

The program described in this section was used to generate Tables 13.1 and 13.2. The program determines the minimal sample size needed to achieve a specified goal with a specified power, when flipping a single coin.

In principle, we could simulate random samples of values from the generating distribution, and then simulate random samples of coin flips based on those particular values of θ. For each repetition, we would tally whether the goal was achieved. With a large number of repetitions, this procedure would provide a good approximation to the power.

In the present simple situation, however, we can determine mathematically the exact probabilities of each possible outcome. The simulated data are generated by sampling a θ value according to our data-generating prior, and then generating N flips of the coin according to the binomial distribution. Therefore the probability of getting z heads, across repeated sampling from the prior, is

$$p(z|N) = \int_0^1 d\theta\ p(z|N,\theta)\, p(\theta)$$

$$
\begin{aligned}
&= \int_0^1 d\theta \; \text{binomial}(z|N,\theta) \; \text{Bernoulli}(\theta|a,b) \\
&= \int_0^1 d\theta \; \binom{N}{z} \theta^{z}(1-\theta)^{(N-z)} \theta^{(a-1)}(1-\theta)^{(b-1)}/B(a,b) \\
&= \binom{N}{z} B(z+a, N-z+b)/B(a,b) \qquad (13.1)
\end{aligned}
$$

(This probability of possible data is sometimes called the "posterior marginal distribution of z"; cf. Eqn. 5 of Pham-Gia & Turkkan, 1992) For each possible outcome, z, we update the audience-agreeable prior to render an audience-agreeable posterior distribution, and then we decide whether the goal has been achieved for that outcome. Because the decision is determined by the outcome, z, the probability of the decisions is determined by the probability of the outcomes. Equation 13.1 is implemented in the program on lines 20–22, but in logarithmic form to prevent under ow errors.

The function minNforHDIpower has several arguments, most of which have default values. Two arguments without defaults are the mean, genPriorMean, and e ective sample size, genPriorN, of the data-generating hypothesis, which is assumed to be a beta density. For example, suppose that the experimenter believes that the bias of a coin is very nearly .65, and suppose that the belief is based on the equivalent of 2000 of ctitious ips of the coin; then genPriorMean = .65 and genPriorN = 2000.

The next two arguments, HDImaxwid and nullVal, specify the goal of the experiment. One and only one of the arguments must be non-NULL. For example, if the goal is for the HDI to exclude the value of 0.5, then nullVal = .5 (and HDImaxwid = NULL). If the goal is for the HDI to have a maximal width of .2, then HDImaxwid = .2 (and nullVal = NULL). The next argument speci es the limits of the ROPE, if relevant. The function in its present form checks only if the HDI excludes the null value, or ROPE if speci ed. The function does not check whether the ROPE fully contains the HDI.

Finally, the function also has arguments for the audience-agreed prior that is used in the Bayesian analysis. It defaults to a uniform prior, but a di erent audience-agreed prior can be speci ed in terms of the mean (audPriorMean) and certainty (audPriorN) of a beta distribution.

Here is an example of how to use the program:

```
source("minNforHDIpower.R")
minNforHDIpower( genPriorMean=.65 , genPriorN=2000 , nullVal=.5 )
```

The program will compute power for increasing values of sample size, until stopping at N=82 (as shown in Table 13.1).

(minNforHDIpower.R)

```
1  minNforHDIpower = function( genPriorMean , genPriorN ,
2                              HDImaxwid=NULL , nullVal=NULL , ROPE=c(nullVal,nullVal) ,
3                              desiredPower=0.8 , audPriorMean=0.5 , audPriorN=2 ,
4                              HDImass=0.95 , initSampSize=20 , verbose=T ) {
5      if ( is.null(HDImaxwid) + is.null(nullVal) != 1 ) {
6          stop("One and only one of HDImaxwid and nullVal must be speci fied.")
7      }
8      # Convert prior mean and N to a,b parameter values of beta dist ribution.
9      genPriorA = genPriorMean * genPriorN
10     genPriorB = ( 1.0 - genPriorMean ) * genPriorN
11     audPriorA = audPriorMean * audPriorN
12     audPriorB = ( 1.0 - audPriorMean ) * audPriorN
13     # Initialize loop for incrementing sampleSize
```

```
    sampleSize = initSampSize
    notPowerfulEnough = TRUE
    # Increment sampleSize until desired power is achieved.
    while( notPowerfulEnough ) {
        zvec = 0:sampleSize # All possible z values for N flips.
        # Compute probability of each z value for data-generating pr        ior.
        pzvec = exp( lchoose( sampleSize , zvec )
                     + lbeta( zvec + genPriorA , sampleSize-zvec + genPriorB )
                     - lbeta( genPriorA , genPriorB ) )
        # For each z value, compute HDI. hdiMat is min, max of HDI for ea    ch z.
        hdiMat = matrix( 0 , nrow=length(zvec) , ncol=2 )
        for ( zIdx in 1:length(zvec) ) {
            z = zvec[zIdx]
            hdiMat[zIdx,] = HDIofICDF( qbeta , shape1 = z + audPriorA ,
                                       shape2 = sampleSize - z + audPriorB )
        }
        hdiWid = hdiMat[,2] - hdiMat[,1]
        if ( !is.null( HDImaxwid ) ) {
            powerHDI = sum( pzvec[ hdiWid < HDImaxwid ] )
        }
        if ( !is.null( nullVal ) ) {
            powerHDI = sum( pzvec[ hdiMat[,1] > ROPE[2] | hdiMat[,2] < RO    PE[1] ] )
        }
        if ( verbose ) {
            cat( " For sample size = ", sampleSize , ", power = " , powerHDI ,
                 "\n" , sep="" ) ; flush.console()
        }
        if ( powerHDI > desiredPower ) {
            notPowerfulEnough = FALSE
        } else {
            sampleSize = sampleSize + 1
        }
    } # End while( notPowerfulEnough )
    # Return the minimal sample size that achieved the desired po      wer.
    return( sampleSize )
} # end of function
```

13.6.2 Power and sample size for multiple mints

This section explains how to use a BUGS model to estimate power for a given sample size. The first step is to create a working BRugs program for a single set of data, as if the data were from an actual experiment. Make sure that the program has appropriate burn-in and thinning so that the posterior sample is robust and useful. (For a reminder of the issues of burn-in and thinning, see Section 23.2, p. 510.) Once a working BRugs program is established, it is modified and wrapped in a function that is called repeatedly with different simulated sets of data.

The general scheme for estimating power with an BRugs program is outlined in Figure 13.3. At the top of the code is the BUGS model wrapped in an R function called GoalAchievedForSample. The function takes simulated data as input, and returns a TRUE/FALSE value indicating whether or not the goal was achieved. As indicated in Figure 13.3, the BUGS model specification remains unchanged. The data specification is modified, however, to accept the passed-in simulated data rather than the actual data. The MCMC chains are then generated as in the original program, using the burn-in and thinning that is already known to produce well-mixed chains. Finally, when the chains are examined, new commands are included to test whether the goal has been achieved. For example, the code could

```
GoalAchievedForSample = function( simulatedData )  {
  library(BRugs) # needed inside function to re-initialize.
  #-------------------------------------------------  ----------
  # Model specification:  Unchanged.
  #-------------------------------------------------  ----------
  # Data:  NEW COMMANDS for using simulatedData.
  #-------------------------------------------------  ----------
  # Initialize and run chains:  Unchanged.
  #-------------------------------------------------  ----------
  # Examine chains:  NEW COMMANDS for checking whether goal is achieved.
  # Denote result by true/false variable named goalAchieved.
  return( goalAchieved )
}

nSimulatedExperiments = 500 # An arbitrary large number.
nSuccess = 0 # Initialize counter.
for ( experimentIdx in 1:nSimulatedExperiments )  {
  simulatedData = ...  # Create simulated data
  nSuccess = nSuccess + GoalAchievedForSample( simulatedData )
  estPower = nSuccess / experimentIdx
}
```

Figure 13.3: Outline of R code for computing power. This outline assumes that there is already a working BRugs script for a single set of data. The working BRugs script is wrapped in a function at the beginning of the code shown here.

check whether the 95% HDI excludes the ROPE. The result is a logical value denoted by the variable `goalAchieved`, which gets returned by the function.

The lower section of code in Figure 13.3 calls the function defined in the upper section of code. The idea is that we will call the function many times, each time with different simulated data, and check whether the goal is achieved. The proportion of times that the goal is achieved is the estimated power. The number of times to call the function is denoted `nSimulatedExperiments`. Then a `for` loop tallies the number of successes. As the number of simulated experiments increases, the estimate becomes more stable.

We can think of each time through the loop, with a new simulated data set, as the flip of a coin that has a probability of heads equal to the true power of the experiment. We are estimating the bias in that coin, i.e., the power of the experiment, by the observed number of successes. If our prior belief about the power is uniform, then the posterior belief is distributed as `beta( nSuccess+1, nSimExperiments-nSuccess+1 )`. The HDI of the estimated power can be computed by calling the function `HDIofICDF( qbeta , shape1 = nSuccess+1 , shape2 = nSimExperiments-nSuccess+1 )`.

A complete instantiation of the scheme is shown in the program listing below. It was used to generate Figure 13.2 and the other results mentioned in Section 13.3. The code is modified from the program used to analyze the real data, listed on p. 188. The program has many embellishments beyond the skeleton outlined in Figure 13.3.

The function, that takes a set of data as input and returns whether or not the goal was achieved, spans lines 5–147 of the program. The function has a couple extra arguments

that help with plotting graphs of the results, which we might want to see for at least a few sets of data to be sure that the simulations are behaving properly. One extra argument is `plotResults`, which is simply a true/false flag indicating whether or not to plot the results. Another extra argument is the filename to be used for saving the plots.

The data section, lines 44–49, is simpler than in the original program, because the complete `dataList` is provided from outside the function. The data section here merely sends the `dataList` to BUGS.

In the results sections, beginning at line 89, there are three main changes. First, the chains are not checked for convergence and mixing, because it is assumed that the burn-in and thinning have already been checked for adequacy. (For a reminder of the issues of burn-in and thinning, see Section 23.2, p. 510.) Second, the plot-producing code (lines 104–132) is wrapped in an `if` statement, which causes plots to be produced and saved only if the argument `plotResults` is true.

The third main change is checking for whether the goal was achieved, as computed in lines 134–144. The code checks for three goals regarding the ${}_c$ values: (i) Does the HDI on filtration minus condensation exceed zero? (ii) Does the HDI on the difference between the two filtration conditions exceed zero? (iii) Is the HDI on the difference of condensation condensations of width less than 0.2? The true/false values of these three goals are returned as a vector.

The function is called by code starting at line 150. The code includes the option for computing either retrospective or replication power, which is specified on line 152. The program then loads the original data, which are used for computing replication power, and the actual posterior sample, which is used for either type of power.

Various constants are specified in lines 170–176. In particular, the simulated number of subjects per condition is specified in line 170. Also notice that the number of goals computed by the function is specified in line 176. The number of goals is used only to initialize the success counter in line 188.

Because we are simulating results from randomly generated sets of data, and each data set may take a few minutes to run, the complete batch of hundreds of data sets may take many hours to complete. Therefore we may want to run only a few simulated experiments at a time, or interrupt a run, and save the interim results for future continuation. Also, be warned that BUGS often decides to crash for no apparent reason on some simulated data sets, and you will find yourself with interrupted runs whether you intended it or not. For these reasons, the program saves the interim results after every simulated data set, on lines 241–242. When the program is invoked, it checks whether there is a previously saved interim result, on lines 178–190, and restores those results if they exist.

The loop that repeatedly generates simulated data sets and the Bayesian analysis begins on line 193. The simulated data are generated in lines 195–209. It uses the `mu` and `kappa` values from the actual posterior to generate simulated data in each condition. The simulated data are assembled into a `dataList` in lines 211–229. If retrospective power is being computed, then only the newly simulated data are included in the `dataList`. If replication power is being computed, then the original, actual data are concatenated onto the simulated data. By including the actual data, the model is effectively using the actual posterior as the prior for the new simulated data.

(FilconBrugsPower.R)

```
1 graphics.off()
2 rm(list=ls(all=TRUE))
3
```

```
4
5  GoalAchievedForSample = function( datalist , plotResults =F ,
6                                    fileNameRoot="DeleteMe" ) {
7
8  library(BRugs)          # Kruschke, J. K. (2010). Doing Bayesian dat a analysis:
9                            # A Tutorial with R and BUGS. Academic Press / Elsevier.
10 #------------------------------------------------- ----------------------------
11 # THE MODEL.
12
13 modelstring = "
14 # BUGS model specification begins here...
15 model {
16     for ( subjIdx in 1:nSubj ) {
17         # Likelihood:
18         z[subjIdx] ~ dbin( theta[subjIdx] , N[subjIdx] )
19         # Prior on theta: Notice nested indexing.
20         theta[subjIdx] ~ dbeta( a[cond[subjIdx]] , b[cond[subjId x]] )I(0.001,0.999)
21     }
22     for ( condIdx in 1:nCond ) {
23         a[condIdx] <- mu[condIdx] * kappa[condIdx]
24         b[condIdx] <- (1-mu[condIdx]) * kappa[condIdx]
25         # Hyperprior on mu and kappa:
26         mu[condIdx] ~ dbeta( Amu , Bmu )
27         kappa[condIdx] ~ dgamma( Skappa , Rkappa )
28     }
29     # Constants for hyperprior:
30     Amu <- 1
31     Bmu <- 1
32     Skappa <- pow(meanGamma,2)/pow(sdGamma,2)
33     Rkappa <- meanGamma/pow(sdGamma,2)
34     meanGamma <- 10
35     sdGamma <- 10
36 }
37 # ... end BUGS model specification
38 " # close quote for modelstring
39 # Write model to a file:
40 writeLines(modelstring,con="model.txt")
41 # Load model file into BRugs and check its syntax:
42 modelCheck( "model.txt" )
43
44 #------------------------------------------------- ----------------------------
45 # THE DATA.
46
47 # datalist supplied from outside the function.
48 # Get the data into BRugs:
49 modelData( bugsData( datalist ) )
50
51 #------------------------------------------------- ----------------------------
52 # INTIALIZE THE CHAINS.
53
54 nChain = 3
55 modelCompile( numChains=nChain )
56
57 if ( F ) {
58     modelGenInits() # often won't work for diffuse prior
59 } else {
60   #  initialization based on data
61   genInitList <- function() {
62     sqzData = .01+.98*datalist$z/datalist$N
```

```
63        mu = aggregate( sqzData , list(datalist$cond) , mean )[,"x"          ]
64        sd = aggregate( sqzData , list(datalist$cond) , sd )[,"x"]
65        kappa = mu*(1-mu)/sd^2 - 1
66        return(
67          list(
68            theta = sqzData ,
69            mu = mu ,
70            kappa = kappa
71          )
72        )
73      }
74      for ( chainIdx in 1 : nChain ) {
75        modelInits( bugsInits( genInitList ) )
76      }
77    }
78
79    #------------------------------------------------                    ----------------------------
80    # RUN THE CHAINS.
81
82    burninSteps = 1000
83    modelUpdate( burninSteps )
84    cat("Got through burn in...");flush.console()
85    samplesSet( c("mu","kappa","theta","a","b") )
86    nPerChain = ceiling(3000/nChain)
87    modelUpdate( nPerChain , thin=5 )
88
89    #------------------------------------------------                    ----------------------------
90    # EXAMINE THE RESULTS.
91
92    # Extract chain values from BUGS:
93    mu = NULL
94    kappa = NULL
95    for ( condIdx in 1:nCond ) {
96          nodeName = paste( "mu[" , condIdx , "]" , sep="" )
97          mu = rbind( mu , samplesSample( nodeName ) )
98          nodeName = paste( "kappa[" , condIdx , "]" , sep="" )
99          kappa = rbind( kappa , samplesSample( nodeName ) )
100   }
101   chainLength = NCOL(mu)
102
103   # Display results if desired:
104   if ( plotResults ) {
105     # Histograms of condition (i.e. group) mu differences:
106     windows(12,4)
107     layout( matrix(1:3,nrow=1) )
108     source("plotPost.R")
109     histInfo = plotPost( mu[1,]-mu[2,] , xlab=expression(mu[          1]-mu[2]) ,
110                          compVal=0.0 , breaks=30 , main="" )
111     histInfo = plotPost( mu[3,]-mu[4,] , xlab=expression(mu[          3]-mu[4]) ,
112                          breaks=30 , main="" )
113     HDIlim = HDIofMCMC( mu[3,]-mu[4,] )
114     text( mean(HDIlim) , .25*max(histInfo$density) , adj=c(.          5,0) , cex=1.25 ,
115           bquote( "HDI width = " * .(signif(HDIlim[2]-HDIlim[1],3))          ) )
116     nSubjPerCond = round( datalist$nSubj / datalist$nCond )
117     histInfo = plotPost( (mu[1,]+mu[2,])/2 - (mu[3,]+mu[4,])          /2 , compVal=0.0 ,
118                          xlab=expression((mu[1]+mu[2])/2 - (mu[3]+mu[4])/2) ,
119                          breaks=30 , main="" )
120     dev.copy2eps( file = paste( fileNameRoot,"N",nSubjPerCo       nd,"_",expIdx,".eps" ,
121                                 sep="" ) )
```

```
  # Scatterplot of mu, kappa:
  windows()
  muLim = c(.60,1) ; kappaLim = c( 2.0 , 50 ) ; mainLab="Posterior"
  thindex = round( seq( 1 , chainLength , len=300 ) )
  plot( mu[1,thindex] , kappa[1,thindex] , main=mainLab ,
        xlab=expression(mu[c]) , ylab=expression(kappa[c]) , cex.lab=1.75 ,
        xlim=muLim , ylim=kappaLim , log="y" , col="red" , pch="1" )
  points( mu[2,thindex] , kappa[2,thindex] , col="blue" , pch="2" )
  points( mu[3,thindex] , kappa[3,thindex] , col="green" , pch="3" )
  points( mu[4,thindex] , kappa[4,thindex] , col="black" , pch="4" )
} # end if plotResults

# Specify goals and check whether they are achieved:
source("HDIofMCMC.R")
# Goal is filtration vs condensation 95% HDI exceeds zero:
goal1Ach = ( HDIofMCMC( (mu[1,]+mu[2,])/2 - (mu[3,]+mu[4,])/2 )[1] > 0.0 )
# Goal is filtration1 vs filtration2 95% HDI exceeds zero:
goal2Ach = ( HDIofMCMC( mu[1,]-mu[2,] )[1] > 0.0 )
# Goal is condensation1 vs condensation2 95% HDI has width less than 0.2:
HDIlim = HDIofMCMC( mu[3,]-mu[4,] )
HDIwidth = HDIlim[2] - HDIlim[1]
goal3Ach = ( HDIwidth < 0.2 )
goalAchieved = c( goal1Ach , goal2Ach , goal3Ach )

return( goalAchieved )
} # end of function GoalAchievedForSample

#==============================================================================
# Now call the function defined above, using simulated data.

analysisType = c("Retro","Repli")[1] # specify [1] or [2]
fileNameRoot = paste("FilconBrugsPower",analysisType,sep="")

# Load original data, for use in replication probability analysis:
# (These lines intentionally exceed the margins so that they don't take up
# excessive space on the printed page.)
CondOfSubjOrig = c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2
nTrlOfSubjOrig = c(64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,
nCorrOfSubjOrig = c(45,63,58,64,58,63,51,60,59,47,63,61,60,51,59,45,61,59,60,58,63,56,63,64,64,60,64
nSubjOrig = length(CondOfSubjOrig)
nCondOrig = length(unique(CondOfSubjOrig))

# Load previously computed posterior mu[cond,step], kappa[cond,step] chains.
load( file="FilconBrugsMuKappa.Rdata" )
chainLength = NCOL(mu)
nCond = NROW(mu) # should be 4, of course

# SPECIFY NUMBER OF SUBJECTS PER GROUP FOR SIMULATED DATA:
nSubjPerCond = 15

# Specify number of simulated experiments:
nSimExperiments = min(200,chainLength)
nSubj = nSubjPerCond * nCond # Number of subjects total.
nTrlPerSubj = 64 # Number of trials per subject; fixed by design at 64.
nGoal=3 # Determined in function above.

# If previous record of power estimation exists, load it and continue the runs.
filelist = dir( pattern=paste(fileNameRoot,"N",nSubjPerCond,"Result.Rdata",sep="") )
if ( length( filelist ) > 0 ) { # if the file already exists...
```

```
    # load  previous expIdx, nSuccess
    load(paste(fileNameRoot,"N",nSubjPerCond,"Result.Rd      ata",sep=""))
    prevExpIdx = expIdx
    # Use just some of the MCMC steps, distributed among the whole   chain:
    chainIdxVec = round(seq(1,chainLength,len=(prevExpIdx      +nSimExperiments)))
} else { # ... otherwise, start a new record
  prevExpIdx = 0
  nSuccess = rep(0,nGoal) # Initialize success counter.
  chainIdxVec = round(seq(1,chainLength,len=nSimExperim    ents))
}

# Simulated experiment loop begins here:
for ( expIdx in (1+prevExpIdx):(nSimExperiments+prevExp      Idx) ) {

    # Generate random data from posterior parameters
    chainIdx=chainIdxVec[expIdx]
    CondOfSubj = sort( rep( 1:nCond , nSubjPerCond ) )
    nTrlOfSubj = rep( nTrlPerSubj , nSubj )
    nCorrOfSubj = rep( 0 , nSubj )
    for ( condIdx in 1:nCond ) {
        m = mu[condIdx,chainIdx]
        k = kappa[condIdx,chainIdx]
        a = m*k
        b = (1-m)*k
        # Generate random theta and z values for simulated subjects:
        thetaVec = rbeta( nSubjPerCond , a , b )
        zVec = rbinom( n=nSubjPerCond , size=nTrlPerSubj , prob=th    etaVec )
        nCorrOfSubj[ CondOfSubj==condIdx ] = zVec
    }

    # Put data into list for BUGS program
    if ( analysisType == "Retro" ) { # retrospective power
        datalist = list(
                nCond = nCond ,
                nSubj = nSubj ,
                cond = CondOfSubj ,
                N = nTrlOfSubj ,
                z = nCorrOfSubj
                )
    }
    if ( analysisType == "Repli" ) { # replication probability
        datalist = list(
                nCond = nCond ,
                nSubj = nSubj + nSubjOrig ,
                cond = c( CondOfSubj , CondOfSubjOrig ) ,
                N = c( nTrlOfSubj , nTrlOfSubjOrig ) ,
                z = c( nCorrOfSubj , nCorrOfSubjOrig )
                )
    }

    # Make plots for first 10 simulated experiments:
    if ( expIdx <= 10 ) { plotRes = T } else { plotRes = F }

    # Call BUGS program and tally number of successes:
    nSuccess = nSuccess + GoalAchievedForSample( datalist ,
                                                 plotRes , fileNameRoot )
    estPower = nSuccess / expIdx
    cat( "\n*** nSubjPerCond:",nSubjPerCond, ", nSimExp:",e      xpIdx,
      " , nSuccess:",nSuccess, ", estPower:",round(estPower,2      ), "\n\n" )
```

```
        flush.console()
        save( nSuccess , expIdx , estPower ,
              file=paste(fileNameRoot,"N",nSubjPerCond,"Result.Rd      ata",sep="") )

} # end of simulated experiment loop.
```

13.7 Exercises

Exercise 13.1. [Purpose: Comic relief.] Read the complete oeuvre of Friedrich Nietzsche, with special attention to his posthumous work, The Will to Power (Nietzsche, 1967). Provide a mathematical formalization of the Nietzschian concept of will and power, using Bayesian probability theory. Show that the notion of statistical power is a special case of formalized Nietzschian power, and vice versa. Write your answer in both English and German. Do not submit your answer to the instructor; do post your answer on your personal web blog.

Exercise 13.2. [Purpose: Understanding power for ipping a single coin, in Tables 13.1 and 13.2.] For this exercise, consider ipping a single coin and inferring its bias.

(A) Table 13.2 indicates that when the data-generating distribution is vague, with = 10 and = :80, then 85 ips are needed for an 80% chance of getting the 95% HDI width to be less than .2. What is the minimal N needed if the data-generating distribution is very certain, with = 2000? Show the command you used, and report the exact power for the smallest N that has power greater than .8.

(B) Regarding the previous part, why might a researcher pursue a goal of precision if the data-generating hypothesis is already very precise? (Hint: The audience prior may be di erent than the data-generating hypothesis. Discuss briefly, perhaps with an example.)

(C) Table 13.1 indicates that when the data-generating distribution is highly certain, with = 2000 and = :80, then 18 ips are needed for an 80% chance of getting the 95% HDI to exclude = :5. What is the minimal N needed if the data-generating distribution is vague, with = 2? Show the command you used, and report the exact power for the smallest N that has power greater than .8.

(D) For the previous part, the goal was for the HDI to exclude the null value (i.e., 0.5). Notice that the goal can be satis ed if the the HDI is above the null value or if the HDI is below the null value. (i) When the data-generating prior is a beta distribution with = 0:8 and = 2, as in the previous part, what proportion of the data-generating biases are greater than the null value? (ii) If the goal is for the HDI to fall entirely above the null value, what sample size is needed to achieve a power of .8? Hint: Use minNforHDIpower.R with the argument ROPE=c(0,.5). Watch the sample size increase and increase and increase, as the power creeping toward an asymptote. Why does the power never exceed the proportion you computed for (i)?

Exercise 13.3. [Purpose: Power determination for groups of coins, when goal is precision.] Consider the ltration-condensation experiment summarized in Section 13.3. Suppose we want the 95% HDI on the di erence, $_3$ $_4$, to have a width less than 0.20. What sample size (N per group) is needed to achieve this goal at least 80% of the time? Determine the answer by running the program in Section 13.6.2 (FilconBrugsPower.R) with various values

for nPerGroup. Hint: N 17; your job is to nd the minimal N and discuss how you did it.

Exercise 13.4. [Purpose: This is a capstone exercise that uses real data to review many techniques of the previous chapters, including generating priors in BUGS, checking credibility of null values, estimating retrospective power, and conducting a posterior predictive check.]

This exercise examines a learning experiment that investigated how easy it is for people to learn new category structures after having previously learned an initial structure (Kruschke, 1996). Some new structures had relevant stimulus dimensions that were also previously relevant in the initial structure, while other new structures had relevant stimulus dimensions that were previously irrelevant in the initial structure. The initial structure and the subsequently learned structures, are outlined in Table 13.4.

Table 13.4: Design of relevance shift experiment reported by Kruschke (1996). Cells indicate category assignment (X or O) of each stimulus. Learners were trained in the Initial Phase, and then seamlessly continued into one of the Reversal, Relevant, Irrelevant or Compound Phases.

	Stimulus							
Phase	j	j	j	j	j	j	j	j
Initial	X	O	O	X	X	O	O	X
Reversal	O	X	X	O	O	X	X	O
Relevant	X	X	O	O	X	X	O	O
Irrelevant	X	X	X	X	O	O	O	O
Compound	X	O	X	O	O	X	O	X

Note. The actual stimuli had somewhat different proportions than the ones displayed here. The assignment of physical stimuli to abstract structural items was randomly permuted for each subject.

The stimuli were simple pictures of freight-train box cars that had three dimensions: position of "door" (left or right), height of box (short or tall), and color of "wheels" (blank or lled). The initial phase involved a structure in which two dimensions are relevant. For example, in Table 13.4, the initial phase can be described as "If it is short and left-doored, or tall and right-doored, then it's an X". Notice that wheel color is irrelevant in the initial phase. After learning that classi cation accurately, people would be trained on one of the four shifts listed in Table 13.4. The Reversal shift reverses all the labels. The Relevant shift uses just one of the previously relevant dimensions. The example in Table 13.4 can be described as "If it's short, then it's an X". The Irrelevant shift uses the one previously irrelevant dimension; the example in Table 13.4 can be described as "If it's blank-wheeled, then it's an X". Finally, the Compound shift requires attention to two dimensions, one of which was previously relevant and one of which was previously irrelevant. The example in Table 13.4 can be described as "If it's blank-wheeled and left-doored, or lled-wheeled and right-doored, then it's an X".

Various theories of learning predict different degrees of difficulty in learning the new structures after the old structures (see p. 230 of Kruschke, 1996). For example, if learn-

ers simply memorize the eight stimuli and their assignments, then the Reversal requires all eight associations to be changed, whereas Relevant, Irrelevant, and Compound shifts all require only four assignments to be changed. Therefore, the reversal should be most difficult to learn, and the other three changes should be equally easy to learn. One of the novel contributions of the experiment was the shift design that allowed direct comparison of "reversal" shift with "intradimensional" and "extradimensional" shifts, which had not been done previously. Another novel contribution was that the shift phase had dimensions that were relevant to the outcome without being individually related with the outcome. Variations of the design have subsequently been used by other researchers (e.g., D. N. George & Pearce, 1999; Oswald et al., 2001).

For each shift, the number of correct responses in the first 32 trials was recorded for each learner. These data are shown in the left column of Figure 5.13, p. 286, and are included in the file Kruschke1996CSdatsum.Rdata. It appears that Reversal shift is easiest (i.e., has highest accuracy), followed by Relevant, Irrelevant, and Compound shifts. The Bayesian analysis will tell us the credibility of those apparent differences between conditions.

(A) [Purpose: Create a BUGS model that has an estimated hyperprior on $_c$ and on $_c$.] In the hierarchical diagram on the right side of Figure 9.17, p. 302, the $_c$ parameters are drawn from a gamma distribution that has its parameters estimated rather than fixed. For our new model, we want to do the analogous estimation for the $_c$ parameters also. Instead of setting A and B as constants, they will be estimated hyperparameters, defined as $a = m \quad k + 1$ and $b = (1 \quad m) \quad k + 1$, with $m \quad \text{dunif}(0;1)$ and $k \quad \text{dgamma}(1:1)$. Here is one way to specify the model: (Kruschke1996CSbugs.R)

```
model {
    for ( i in 1:nSubj ) {
        nCorrOfSubj[i] ~ dbin( theta[i] , nTrlOfSubj[i] )
        theta[i] ~ dbeta( a[ CondOfSubj[i] ] , b[ CondOfSubj[i] ] )I(              0.0001,0.9999)
    }
    for ( cond in 1:nCond ) {
        a[cond] <- mu[cond] * kappa[cond]
        b[cond] <- (1-mu[cond]) * kappa[cond]
        mu[cond] ~ dbeta( aMu , bMu )
        kappa[cond] ~ dgamma( sGamma , rGamma )
    }
    aMu <- max( .01 , mMu * kMu )
    bMu <- max( .01 , (1-mMu) * kMu )
    mMu ~ dunif(0,1)
    kMu ~ dgamma(1,.1)
    sGamma <- max( .005 , pow(muGamma,2)/pow(sigmaGamma,2) )
    rGamma <- max( .005 , muGamma/pow(sigmaGamma,2) )
    muGamma ~ dgamma(1,.1)
    sigmaGamma ~ dgamma(1,.1)
}
```

Discuss why we would want to estimate higher-level distributions across the $_c$ and $_c$. You may want to mention commonalities across conditions, such as the learners being the same species, and all learners experiencing the same stimuli on the same computer display. It is also important to discuss shrinkage of estimates. Finally, discuss why the particular higher-level distribution might not be appropriate.

(B) [Purpose: Check for convergence, mixing, and autocorrelation.] With the data included, run the BUGS model and check for convergence, mixing, and autocorrelation. The model can be initialized automatically, using modelGenInits() , but unfortunately takes forever for

the chains to converge, because some are initialized too far away from the mode of the posterior. Instead, manually initialize the chains at reasonable values, as indicated by the data. The same method as was used in Section 9.5.2 (Kruschke1996CSbugs.R) is used and extended here: (Kruschke1996CSbugs.R)

```
genInitList <- function() {
    sqzData = .01+.98*datalist$nCorrOfSubj/datalist$nTrlOfSubj
    mu = aggregate( sqzData , list(datalist$CondOfSubj) , "mean" )[,"x"]
    sd = aggregate( sqzData , list(datalist$CondOfSubj) , "sd" )[,"x"]
    kappa = mu*(1-mu)/sd^2 - 1
    mMu = mean( mu )
    kMu = mMu * (1-mMu) / sd(mu)^2
    muGamma = mean( kappa )
    sigmaGamma = sd( kappa )
    return( list( theta = sqzData ,
                  mu = mu ,
                  kappa = kappa ,
                  mMu = mMu ,
                  kMu = kMu ,
                  muGamma = muGamma ,
                  sigmaGamma = sigmaGamma ) )
}
for ( chainIdx in 1 : nchain ) {
    modelInits( bugsInits( genInitList ) )
}
```

Run the model and determine reasonable burn-in and thinning. Show the autocorrelation and chain plots for $_c$ and $_c$. You may find it useful to refer to Section 23.2, p. 510.

(C) [Purpose: Examine and interpret the posterior distribution of differences.] Which groups are different from each other, on which parameters? Hint: See Figure 13.4.

(D) [Purpose: Check the robustness of the posterior when the prior is changed in reasonable ways.] Do the posterior differences change much if the prior changes? (i) Specifically, try this alternative vague prior: Wherever the top-level prior specifies dgamma(1,.1), change it to dgamma(0.1,.1). (ii) Also, try this prior that forces all $_c$ to be close to 15:

```
muGamma  dgamma(22500,1500)
sigmaGamma  dgamma(25,250)
```

Show your results. Which prior is more reasonable, and why?

(E) [Purpose: Conduct a posterior predictive check.] For this part, use the plausible dgamma(1,.1) prior, not the less-plausible other priors explored in the previous part. Conduct a posterior predictive check by generating simulated data from the sampled posterior parameter values. This can be done in R as follows: (Kruschke1996CSbugs.R)

```
nSimExps = min(500,chainLength) # number of simulated experiments
nSubjPerCond = sum( CondOfSubj == 1 ) # number of subjects per condition
nTrlPerSubj = nTrlOfSubj[1] # number of trials per subject
nCorrOfSubjPredMat = matrix( 0 , nrow=nSimExps , ncol=nSubjPerCond*nCond )
CondOfSubjPredMat =  matrix( 0 , nrow=nSimExps , ncol=nSubjPerCond*nCond )
nTrlOfSubjPredMat =  matrix( 0 , nrow=nSimExps , ncol=nSubjPerCond*nCond )
for ( stepIdx in 1:nSimExps ) {
    for ( condIdx in 1:nCond ) {
        m = mu[condIdx,stepIdx]
        k = kappa[condIdx,stepIdx]
        a = m*k + 1
        b = (1-m)*k + 1
        for ( subjIdx in 1:nSubjPerCond ) {
```

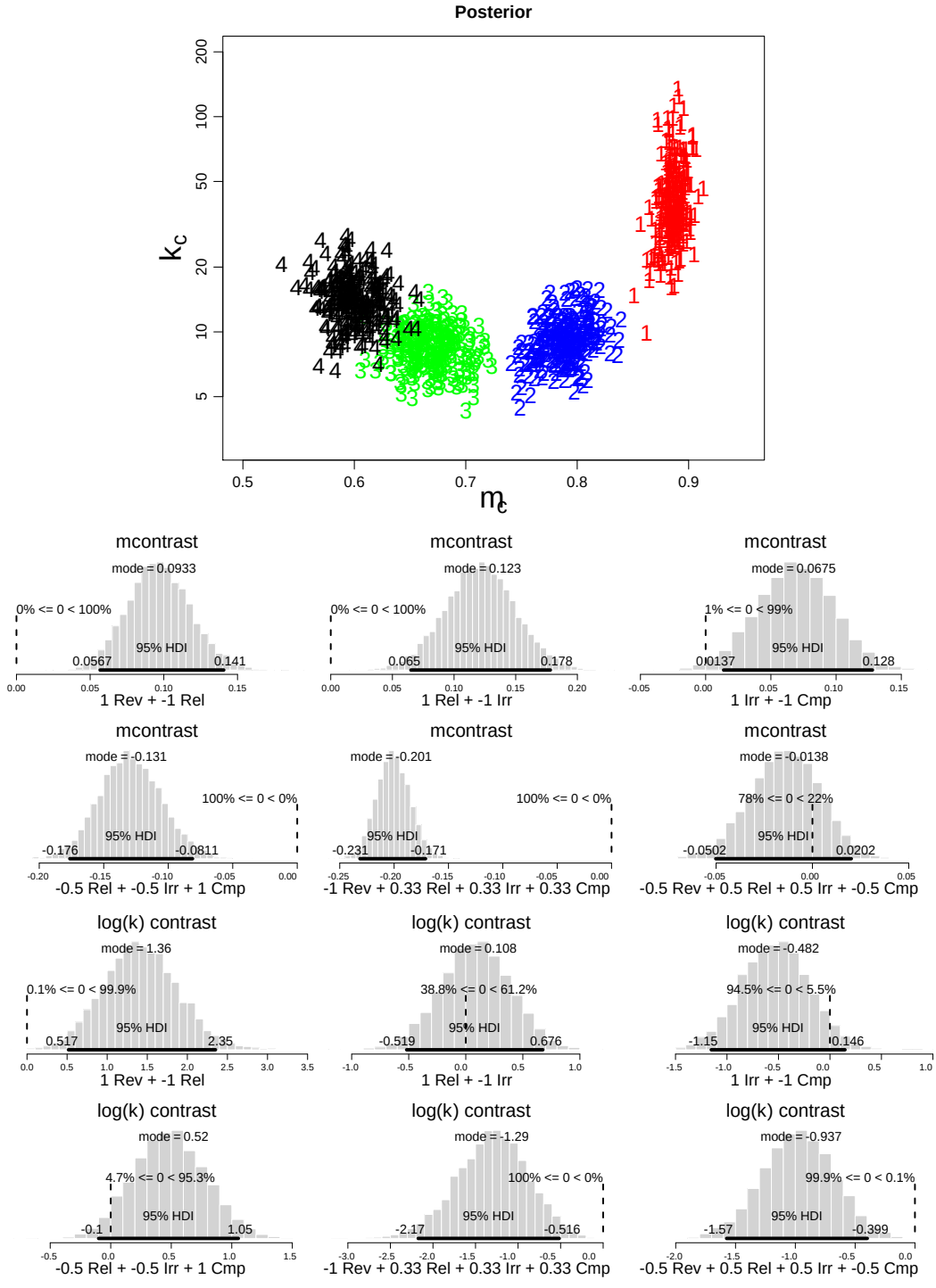

Figure 13.4: For Exercise 13.4, Part C. Posterior and values, and their differences. In the scatterplot, numerals represent a step in the chain for condition.

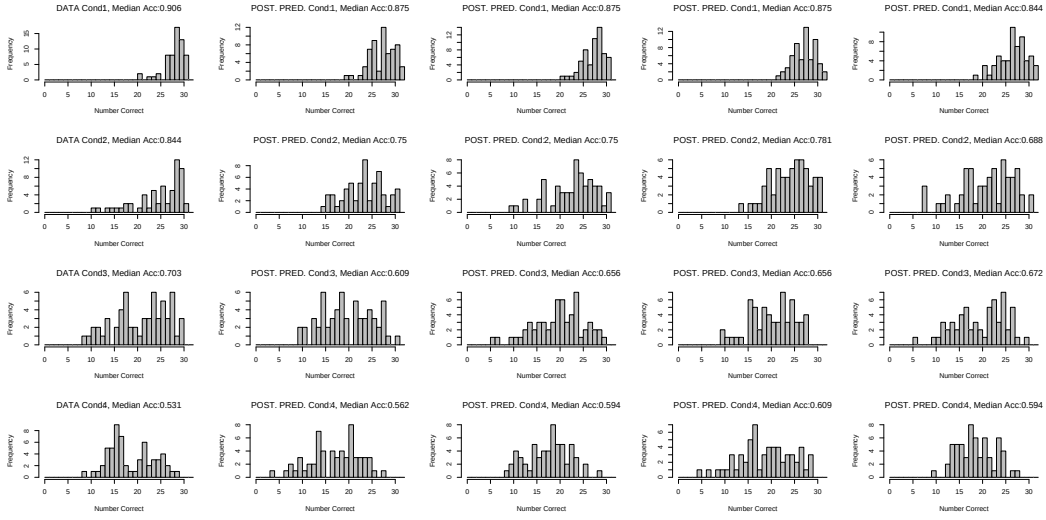

Figure 13.5: For Exercise 13.4, Part E. Actual data in left column, with examples of posterior-predicted data in other columns. Top row is Reversal, second row is Relevant, third row is Irrelevant, and fourth row is Compound. There is a suggestion of bimodality in the data from the Compound Shift (two-left histogram) that is not often generated by simulated posterior-predicted data. Follow-up modeling and empirical work might want to investigate that bimodality if it is deemed theoretically interesting.

```
216                 colIdx = (condIdx-1)*nSubjPerCond + subjIdx
217                 theta = rbeta( 1 , a , b )
218                 nCorrOfSubjPredMat[stepIdx,colIdx] = rbinom( 1 ,
219                                         size=nTrlPerSubj , prob=theta )
220                 CondOfSubjPredMat[stepIdx,colIdx] = condIdx
221                 nTrlOfSubjPredMat[stepIdx,colIdx] = nTrlPerSubj
222             }
223         }
224 }
```

This code, above, puts each randomly generated sample of data in a row of the matrix `nCorrOfSubjPredMat`. Figure 13.5 shows histograms of simulated data. The above code relied on previously extracting the parameter chains from BUGS, as follows: (Kruschke1996CSbugs.R)

```
127 mu = NULL
128 kappa = NULL
129 for ( condIdx in 1:nCond ) {
130     nodeName = paste( "mu[" , condIdx , "]" , sep="" )
131     mu = rbind( mu , samplesSample( nodeName ) )
132     nodeName = paste( "kappa[" , condIdx , "]" , sep="" )
133     kappa = rbind( kappa , samplesSample( nodeName ) )
134 }
```

(F) [Purpose: Do a retrospective power analysis.] Using the method of Figure 13.3, which is exemplified by the code explained in Section 13.6.2 (BernBugsPower.R), conduct a retrospective power analysis, and estimate the power of the goals: $_{Rev} - _{Rel} > 0$, $_{Rel} - _{Irr} > 0$, $_{Irr} - _{Com} > 0$, and $_{Rev} - _{Rel} > 0$ (condition 1 is Reversal, condition 2 is Relevant, condition 3 is Irrelevant, and condition 4 is Compound). To do this, first analyze the actual

data, and get a posterior sample of σ and σ_c values. Then, wrap the BUGS program into a function, into which you can pass data and from which you can check whether each goal is achieved. Then make a loop that goes stepwise through the σ_c, σ chain and generates simulated data from those credible parameter values. At each step, pass the simulated data into the BUGS analysis function. Run at least 100 simulated data sets, and report your tally for how many times the goals were achieved. Cautions and hints: If each BUGS analysis takes two minutes to run, a batch of a few hundred simulated data sets will take hours. Plan accordingly. Sometimes BUGS will run fine for dozens of simulated data sets, and then inexplicably crash on the next random data set. Therefore be sure that your results regarding power estimation are saved at the end of each data set. The goal of showing $\omega_{Rev} - \omega_{Rel} > 0$ has a retrospective power of about 100%, but at least one of the other goals involving σ_c has retrospective power under 50%, which indicates that with 60 subjects per condition, this experiment might have been underpowered. Bayesian analysis does not care whether there are equal numbers of subjects in each condition, so follow-up work could use more subjects in some conditions and fewer subjects in other condtions. The goal of showing $\omega_{Rev} - \omega_{Rel} > 0$ has a retrospective power of only 20%, which is caused largely by shrinkage of σ_c estimates.

(G) [Purpose: Do a replication power analysis.] Suppose you repeat the experiment, using the posterior of the first experiment as the prior for the second experiment. Estimate the probability of achieving these four goals: $\omega_{Rev} - \omega_{Rel} > 0$, $\omega_{Rel} - \omega_{Irr} > 0$, $\omega_{Irr} - \omega_{Com} > 0$, and $\omega_{Rev} - \omega_{Rel} > 0$ (condition 1 is Reversal, condition 2 is Relevant, condition 3 is Irrelevant, and condition 4 is Compound). In other words, conduct an analysis of replication probability, regarding the same goals as the previous part. This task is easy to do if you successfully accomplished the previous part. In each step of the chain, instead of using only the simulated data, concatenate the simulated data to the actual data. Cautions and hints: The runs will take even longer than in the previous part because of the larger data sets.

Part III

The Generalized Linear Model

Chapter 14

Overview of the Generalized Linear Model

Contents

> Straight and proportionate, deep in your core
> All is orthogonal, ceiling to floor.
> But on the outside the vines creep and twist
> 'round all the parapets shrouded in mist.

The previous part of the book explored all the basic concepts of Bayesian analysis applied to a simple likelihood function, namely the Bernoulli distribution. The focus on a simple likelihood function allowed the complex concepts of Bayesian analysis, such as MCMC methods and hierarchical priors, to be developed without interference from additional complications of elaborate likelihood functions with multiple parameters.

In this new part of the book, we apply all the concepts to a more complex but versatile model known as the generalized linear model (GLM; McCullagh & Nelder, 1989; Nelder & Wedderburn, 1972). This model comprises the traditional "off the shelf" analyses such as t-tests, analysis of variance (ANOVA), multiple linear regression, logistic regression, etc. Because we now know from previous chapters the concepts and mechanisms of Bayesian analysis, we can focus on applications of this versatile model. The present chapter is important for understanding subsequent chapters because it lays out the framework for all the models in the remainder of the book.

14.1 The generalized linear model (GLM)

To understand the generalized linear model and its many special cases, we must build up a variety of component concepts regarding relationships between variables and how variables are measured in the first place.

14.1.1 Predictor and predicted variables

Suppose we want to predict someone's weight from their height. In this case, weight is the predicted variable and height is the predictor. Or, suppose we want to predict high school grade point average (GPA) from Scholastic Aptitude Test (SAT) score and family income. In this case, GPA is the predicted variable, while SAT and income are predictor variables. Or, suppose we want to predict the blood pressure of patients who either take drug A, or take drug B, or take a placebo, or merely wait. In this case, the predicted variable is blood pressure, and treatment-group membership is the predictor.

The key mathematical difference between predictor and predicted variables is that the likelihood function expresses the probability of values of the predicted variable as a function of values of the predictor variable. The likelihood function does not describe the probabilities of values of the predictor variable. The value of the predictor variable comes from "outside" the system being modeled, whereas the value of the predicted variable depends on the value of the predictor variable.

Because the predicted variable depends on the predictor variable, at least mathematically in the likelihood function if not causally in the real world, the predicted variable can also be called the "dependent" variable. The predictor variables are sometimes called "independent" variables. The key conceptual difference between independent and dependent variable is that the value of the dependent variable depends on the value of the independent variable. The term "independent" can be confusing because it can be used strictly or loosely. In experimental settings, the variables that are really manipulated and set by the experimenter are the independent variables. In this context of experimental manipulation, the values of the independent variables truly are (in principle at least) independent of the values of other variables, because the experimenter has arranged to arbitrarily set the values of the independent variables. But sometimes a non-manipulated variable is also referred to as "independent", merely as a way to indicate that it is being used as a predictor variable.

Among non-manipulated variables, the roles of predicted and predictor are arbitrary, determined only by the interpretation of the analysis. Consider, for example, people's weights and heights. We could be interested in predicting a person's weight from his/her height, or we could be interested in predicting a person's height from his/her weight. Prediction is merely a mathematical dependency, not necessarily a implication of underlying causal

relationship. Although height and weight tend to co-vary as people, the two variables are not directly causally related. When a person slouches, by getting shorter, he/she does not lose weight. And when a person drinks a glass of water, thereby weighing more, he/she does not get taller.

Just as "prediction" does not imply causation, "prediction" also does not imply any temporal relation between the variables. For example, we may want to predict a person's sex, male or female, from his/her height. Because males tend to be taller than females, on average, this prediction can be made with better than chance accuracy. But a person's sex is not caused by his/her height, nor does a person's sex occur only after their height is measured. Thus, we can "predict" a person's sex from his/her height, but this does not mean that the person's sex occurred later in time than his/her height.

In summary: All manipulated independent variables are predictor variables, not predicted. Some dependent variables can take on the role of predictor variables, if desired. All predicted variables are dependent variables. The likelihood function specifies the probability of values of the predicted variables as a function of the values of the predictor variables.

Why we care. We care about these distinctions between predicted and predictor variables because the likelihood function is a mathematical description of the dependency of the predicted variable on the predictor variable. The thing we have to do in statistical inference is identify what variables we are interested in predicting, on the basis of what predictors.

14.1.2 Scale types: metric, ordinal, nominal

Items can be measured on different scales. For example, the participants in a foot race can be measured either by the time they took to run the race, or by placing in the race (1st, 2nd, 3rd, etc.), or by the name of the team they represent. These three measurements are examples of metric, ordinal, and nominal scales, respectively (Stevens, 1946).

Examples of metric scales include response time (i.e., latency or duration), temperature, height, and weight. Those are actually cases of a special type of metric scale, called a ratio scale, because they have a natural zero point on the scale. The zero point on the scale corresponds to there being a complete absence of the thing being measured. For example, when the duration is zero, there has been no time elapsed, and when the weight is zero, there is no downward force. Because these scales have a natural zero point, it is meaningful to talk about ratios of amounts being measured, and that is why they are called ratio scales. For example, it is meaningful to say that taking 2 minutes to solve a problem is twice as long as taking 1 minute to solve the problem. On the other hand, the scale of historical time has no known absolute zero. We cannot say, for example, that there is twice as much time in January 2nd as there is time in January 1st. We can refer to the duration since some arbitrary reference point, but we cannot talk about the absolute amount of time in any given moment. Scales that have no natural zero are called interval scales because all we know about them is the amount of stuff in an interval on the scale, not the amount of stuff at a point on the scale. Despite the conceptual difference between ratio and interval scales, we will lump them together into the category of metric scales.

A special case of metric-scaled data is count data, also called frequency data. For example, the number of cars that pass through an intersection during an hour is a count. The number of poll respondents who say they belong to a particular political party is a count. Count data can only have values that are non-negative integers. Distances between counts have meaning, and therefore the data are metric, but because the data cannot be negative

and are not continuous, they are treated with different mathematical forms than continuous, real-valued metric data.

Examples of ordinal scales include placing in a race, or rating of degree of agreement. When we are told that, in a race, Jane came in first, Jill came in second, and Jasmine came in third, we only know the order. We do not know whether Jane beat Jill by a nose or by a mile. There is no distance or metric information in an ordinal scale. As another example, many polls have ordinal response scales: Indicate how much you agree with this statement: "Bayesian statistical inference is better than null hypothesis significance testing", with 5 = strongly agree, 4 = mildly agree, 3 = neither agree nor disagree, 2 = mildly disagree, and 1 = strongly disagree. Notice that there is no metric information in the response scale, because we cannot say the difference between ratings of 5 and 4 is the same amount of difference as between ratings of 4 and 3.

Examples of nominal, a.k.a. categorical, scales include political party affiliation, the face of a rolled die, and the result of a flipped coin. For nominal scales, there is neither distance between categories nor order between categories. For example, suppose we measure the political party affiliation of a person. The categories of the scale might be Green, Democrat, Republican, Libertarian, and Other. While some political theories might infer that the parties fall on some underlying liberal-conservative scale, there is no such scale in the actual categorical values themselves. In the actual categorical labels there is neither distance nor ordering.

In summary, if two items have different nominal values, all we know is that the two items are different (and what categories they are in). On the other hand, if items have different ordinal values, we know that the two items are different and we know which one is "larger" than the other, but not how much larger. If two items have different metric values, then we know that they are different, which one is larger, and how much larger.

Why we care. We care about the scale type because the likelihood function must specify a probability distribution on the appropriate scale. If the scale has two nominal values, then a Bernoulli likelihood function may be appropriate. If the scale is metric, then a normal distribution may be appropriate as a likelihood function. Whenever we a choosing a model for data, we must answer the question, What kind of scale are we dealing with?

In the following sections, we will first consider the case of a metric predicted variable with metric predictors. In that context of all metric variables, we will develop the concepts of linear functions and interactions. Once those concepts are established for metric predictors, the notions will be extended to nominal predictors.

14.1.3 Linear function of a single metric predictor

Suppose we have identified one variable to be predicted, which we'll call y, and one variable to be the predictor, which we'll call x. Suppose we have determined that both variables are metric. The next issue we need to address is how to model the relationship between x and y. There are many possible dependencies of y on x, and the particular form of the dependency is determined by the specific meanings and nature of the variables. But in general, across all possible domains, what is the most basic or simplistic dependency of y on x that we might consider? The usual answer to this question is, a linear relationship. A linear function is the generic, "vanilla", off-the-shelf dependency that is used in statistical models. Its methods can be generalized to other models when needed.

Linear functions preserve proportionality. If you double the input, then you double the output. If cost of a book is a linear function of the number of pages, then when the number

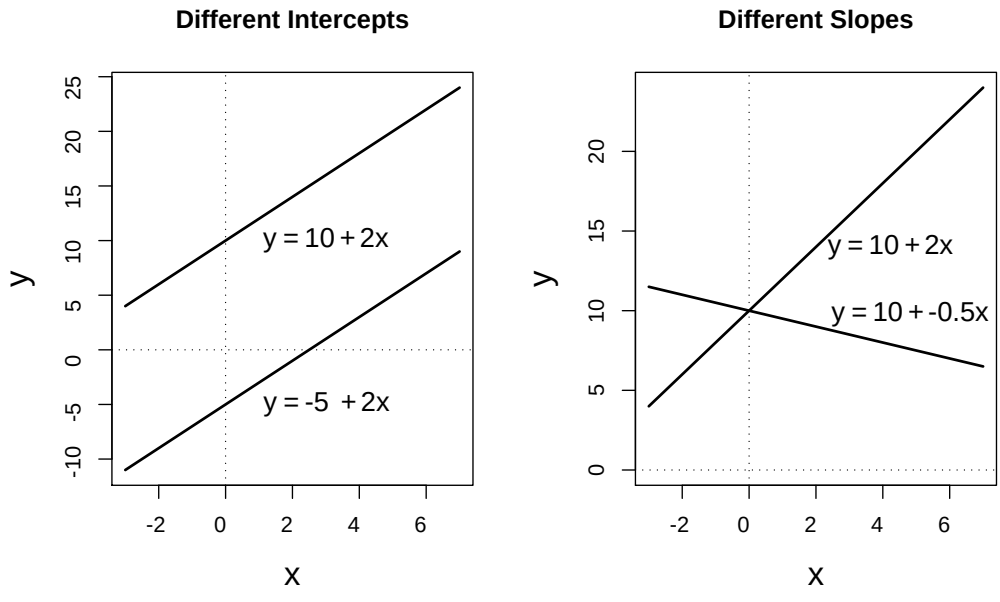

Figure 14.1: Examples of linear functions of a single variable. The left panel shows examples of two lines with the same slope but different intercepts. The right panel shows examples of two lines with the same intercept but different slopes.

of pages is reduced 10%, the cost should be reduced 10%. If automobile speed is a linear function of gas delivery to the engine, then when you press the pedal 20% further, the car should go 20% faster. Non-linear functions do not preserve proportionality. For example, in actuality, car speed is not a linear function of gas delivery. At higher and higher speeds, it takes proportionally more and more gas to make the car go faster. Despite the fact that many real-world dependencies are non-linear, most are at least approximately linear over moderate ranges of the variables. For example, if you have twice the wall area, it takes approximately twice the amount of paint. It is also the case that linear relationships are intuitively prominent (Brehmer, 1974; Hoffman, Earle, & Slovic, 1981; Kalish, Griffiths, & Lewandowsky, 2007). Linear relationships are the easiest to think about: Turn the steering wheel twice as far, and the car should turn twice as sharply. Turn the volume knob 50% higher, the loudness should increase 50%.

The general mathematical form for a linear function of a single variable is

$$y = \beta_0 + \beta_1 x \tag{14.1}$$

When values of x and y that satisfy Equation 14.1 are plotted, they form a line. Examples are shown in Figure 14.1. The value of parameter β_0 is called the y-intercept because it is the where the line intersects the y axis when $x = 0$. The left panel of Figure 14.1 shows two lines with different y-intercepts. The value of parameter β_1 is called the slope because it indicates how much y increases when x increase by 1. The right panel of Figure 14.1 shows two lines with the same intercept but different slopes.

In strict mathematical terminology, the type of transformation in Equation 14.1 is called affine. When $\beta_0 \neq 0$, the transformation does not preserve proportionality. For example, consider $y = 10 + 2x$. When x is doubled from $x = 1$ to $x = 2$, y increases from $y = 12$ to

$y = 14$, which is not doubling y. Nevertheless, the rate of increase in y is the same for all values of x: Whenever x increases by 1, y increases by 2. Equation 14.1 can be algebraically re-arranged so that it does preserve proportionality, as will be shown next.

14.1.3.1 Reparameterization to x threshold form

Equation 14.1 can be algebraically re-arranged as follows:

$$\begin{aligned} y &= \beta_0 + \beta_1 x \\ &= \beta_1 \big(x - \underbrace{(-\beta_0/\beta_1)}_{\theta} \big) \end{aligned} \tag{14.2}$$

This form of the equation is useful because it explicitly shows the value of the x-intercept, a.k.a. threshold, denoted θ. (Do not confuse this use of the symbol θ with different uses in previous chapters.) The threshold is the value of x when y is zero. This is sometimes also called the x intercept.

The x-threshold form preserves proportionality for $x - \theta$. As an example, consider again the case of $y = 10 + 2x$. When changed to x-threshold form, it becomes $y = 2(x + 5)$. When x changes from 1 to 2, $x + 5$ changes from 6 to 7, which is an increase of $(7-6)/6 = 1/6$. The resulting change in y is from 12 to 14, which is an increase of $(14-12)/12 = 1/6$. Thus, a $1/6$ increase in $x - \theta$ results in a $1/6$ increase in y.

The threshold (i.e., x intercept) is often more meaningful than the y-intercept. For example, suppose we are piloting a tugboat upstream on the Mississippi river, and we want to predict how much headway y we will gain against the current for a given setting of the throttle x. Suppose it is the case that $y = -2 + 4x$. This form of the equation indicates that when we apply zero engine power, that is when $x = 0$, then we lose 2 miles an hour, i.e., $y = -2$. In other words, the y intercept tells us the baseline speed of the river current that we are trying to overcome. What may be more useful to know, however, is the amount of engine power we need to apply in order to overcome the current. How big must x be so that we are just matching the downstream pressure? The answer to this question is the threshold, i.e., the value of x that makes $y = 0$. In our example, wherein $y = -2 + 4x$, the threshold is $\theta = -(-2/4) = 0.5$. In other words, when the throttle is set above the threshold of 0.5, then we make progress upstream because $y > 0$, but when the throttle is set below the threshold of 0.5, the we drift downstream because $y < 0$. Thus, the more intuitive form of the "headway" equation is the x-intercept form, $y = 4(x - 0.5)$, because it shows explicitly that our headway is proportional to how much the throttle exceeds 0.5.

Summary of why we care. The likelihood function specifies the form of the dependency of y on x. When y and x are metric variables, the simplest form of dependency, both mathematically and intuitively, is one that preserves proportionality. The mathematical expression of this relation is a so-called linear function. The usual mathematical expression of a line is the y intercept form, but often a more intuitive expression is the x threshold form. Linear functions form the core of most statistical models, it is important to become facile with their algebraic forms and graphical representations.

14.1.4 Additive combination of metric predictors

If we have more than one predictor variable, what function should we use to combine the influences of all the predictor variables? If we want the combination to be linear in each

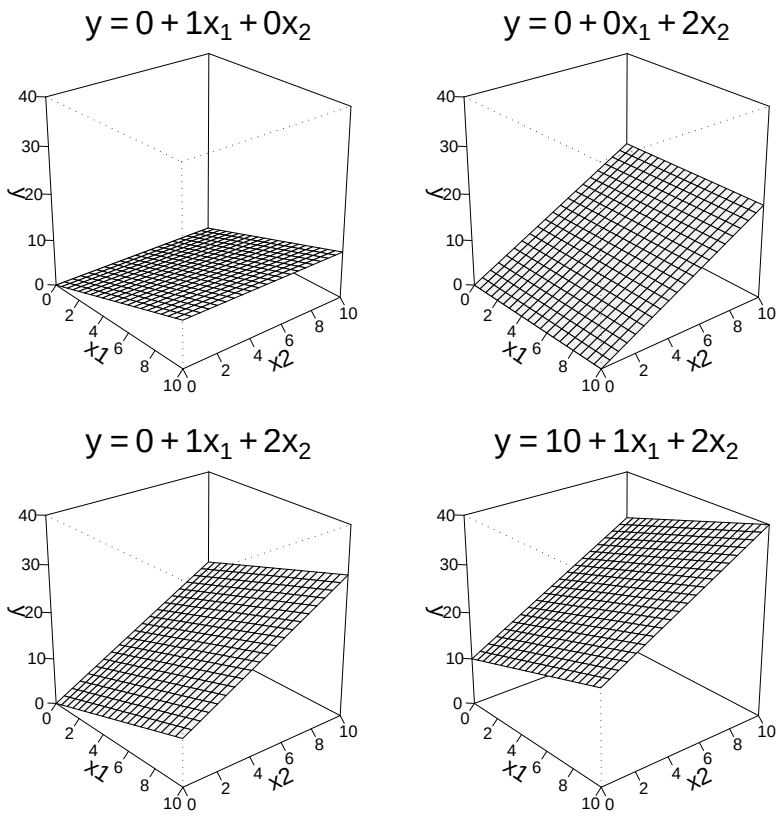

Figure 14.2: Examples of linear functions of two variables, x_1 and x_2. Upper left: Only x_1 has an influence on y. Upper right: Only x_2 has an influence on y. Lower left: x_1 and x_2 have an additive influence on y, compare with upper panels. Lower right: Non-zero intercept is added; compare with lower left panel.

of the predictor variables, then there is just one answer to it. In other words, if we want an increase in one predictor variable to predict the same proportional increase in the predicted variable for any value of the other predictor variables, then the predictions of the individual predictor variables must be added.

In general, a linear combination of K predictor variables has the form

$$\begin{aligned} y &= \beta_0 + \beta_1 x_1 + \cdots + \beta_K x_K \\ &= \beta_0 + \sum_{k=1}^{K} \beta_k x_k \end{aligned} \tag{14.3}$$

Figure 14.2 shows examples of linear functions of two variables, x_1 and x_2. The graphs show y plotted only over a the domain with $0 \le x_1 \le 10$ and $0 \le x_2 \le 10$. It is important to realize that the plane extends from minus to plus infinity, but the graphs only show a small region. Notice in the upper left panel, where $y = 0 + 1x_1 + 0x_2$, that when $x_1 = 10$, then $y = 10$, regardless of the value of x_2. The plane tilts upward in the x_1 direction, but the plane is horizontal in the x_2 direction. The opposite is true in the upper-right panel. The plane tilts upward in the x_2 direction, but the plane is horizontal in the x_1 direction, because there $y = 0 + 0x_1 + 2x_2$. The lower-left panel shows the two influences added together: $y = 0 + 1x_1 + 2x_2$. Notice that the slope in the x_2 direction is steeper than in the x_1 direction. Most importantly,

notice that the slope in the x_2 direction is the same at any specific value of x_1. For example, when $x_1 = 0$, y rises from $y = 0$ to $y = 20$, i.e. an increase of 20, when x_2 goes from $x_2 = 0$ to $x_2 = 10$. And when $x_1 = 10$, y rises from $y = 10$ to $y = 30$, again an increase of 20, when x_2 goes from $x_2 = 0$ to $x_2 = 10$.

14.1.4.1 Reparameterization to threshold form

For notational convenience, define the length of a vector $\vec{\omega} = \langle \omega_1, \ldots, \omega_K \rangle$ to be $|\vec{\omega}| = \left(\sum_k \omega_k^2 \right)^{1/2}$. This may look complicated, but it's merely the everyday formula for distance from the Pythagorean theorem. Carpenters all memorize a special case of this relationship, known as the "3-4-5 rule": When the lengths of the short edges of a right-angled triangle are 3 and 4, then the length of the long edge is exactly 5, because $5 = (3^2 + 4^2)^{1/2}$. (Carpenters actually use the 3-4-5 rule to infer an angle from lengths, rather than infer a third length from two lengths and an angle. Carpenters know that if a triangle has edge lengths of 3, 4, and 5, then the angle between the short edges is exactly 90 degrees.) With this new notation for length, Equation 14.3 can be algebraically re-expressed as

$$\begin{aligned} y &= \omega_0 + \sum_{k=1}^{K} \omega_k x_k \\ &= \omega_0 + |\vec{\omega}| \sum_{k=1}^{K} \frac{\omega_k}{|\vec{\omega}|} x_k \\ &= \omega_0 + |\vec{\omega}| \left(\sum_{k=1}^{K} \frac{\omega_k}{|\vec{\omega}|} x_k \right) \\ &= \omega_0 + |\vec{\omega}| \left(\sum_{k=1}^{K} \cos(\theta_k) x_k \right) \\ &= \omega_0 + |\vec{\omega}| \, |\vec{x}| \cos(\theta) \end{aligned} \tag{14.4}$$

Notice that when there is only a single predictor variable, i.e., when $K = 1$, then $|\vec{\omega}| = |\omega_1|$ and Equation 14.4 reduces to Equation 14.2.

In Equation 14.4, the value of $|\vec{x}|$ is the (Euclidean) length of $\vec{x}$ when $y = 0$ and when $\vec{x}$ is in the direction of vector $\langle \omega_1, \ldots, \omega_K \rangle$. In other words, when $\vec{x} = -\omega_0 \frac{\vec{\omega}}{|\vec{\omega}|^2}$, then $y = 0$. When the length of $\vec{x}$ (in that direction) exceeds the threshold, then $y > 0$. The x threshold form in Equation 14.4 becomes especially useful when we discuss logistic regression in future chapters.

Summary of section: When the influence of every individual predictor is unchanged by changing the values of other predictors, then the influences are additive. The combined influence of two or more predictors can be additive even if the individual influences are nonlinear. But if the individual influences are linear, and the combined influence is additive, then the overall combined influence is also linear. The form of Equation 14.3, or its reparameterization in Equation 14.4, is known as the linear model. It forms the core of many statistical models.

14.1.5 Nonadditive interaction of metric predictors

The combined influence of two predictors does not have to be additive. Consider, for example, a person's self-rating of happiness, predicted from their overall health and annual

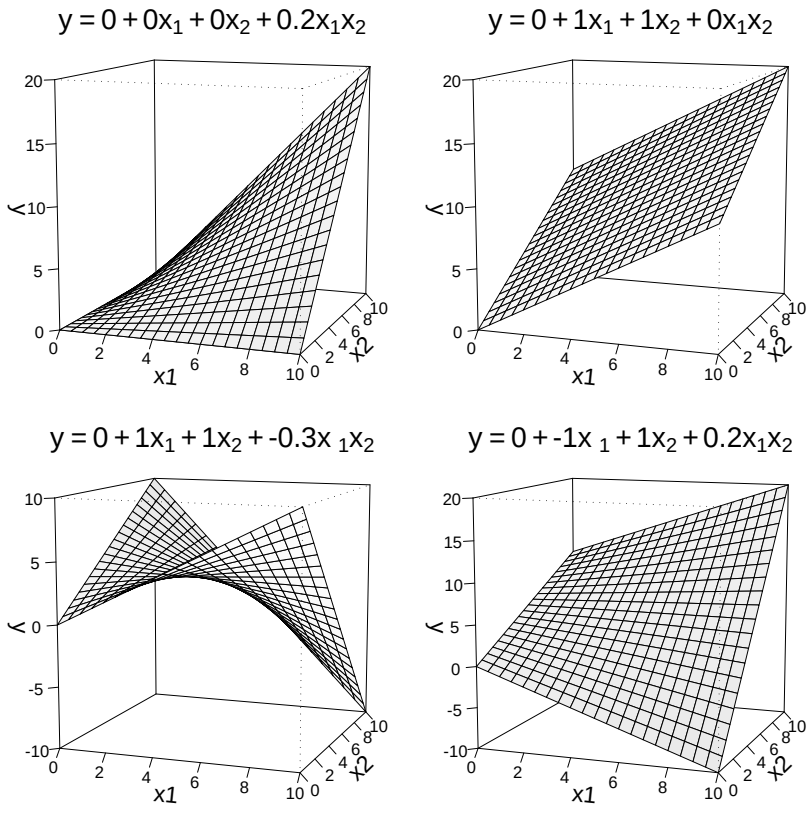

Figure 14.3: Multiplicative interaction of two variables, x_1 and x_2. Upper right panel shows zero interaction, for comparison. Figure 17.3.4, provides additional perspective and insight.

income. It's likely that if a person's health is very poor, then the person is not happy, regardless of his/her income. And if the person has zero income, then the person is probably not happy, regardless of his/her health. But if the person is both healthy and rich, then the person is probably happy (despite celebrated counter-examples in the popular media).

A graph of this sort of non-additive interaction between predictors appears in the upper left panel of Figure 14.3. The vertical axis, labeled y, is happiness. The horizontal axes, x_1 and x_2, are health and income. Notice that if either $x_1 = 0$ or $x_2 = 0$, then $y = 0$. But if both $x_1 > 0$ and $x_2 > 0$, then $y > 0$. The specific form of interaction plotted here is multiplicative: $y = 0 + 0x_1 + 0x_2 + .2x_1x_2$. For comparison, the upper-right panel of Figure 14.3 shows a non-interactive (i.e., additive) combination of x_1 and x_2. Notice that the graph of the interaction has a twist in it, but the graph of the additive combination is flat.

The lower-left panel of Figure 14.3 shows a multiplicative interaction in which the individual predictors increase the outcome, but the combination of the variables decrease the outcome. A real-world example of this occurs with some drugs: Individually, each of two drugs might improve symptoms, but when taken together, the two drugs might interact and cause a decline in health. As another example, consider lighter-than-air travel, i.e., ballooning. The levity of a balloon is increased by fire, as in hot air balloons. And the levity of a balloon is increased by hydrogen, as in many early-20th century blimps and dirigibles. But the levity of a balloon is dramatically decreased by the combination of fire and hydrogen.

The lower-right panel of Figure 14.3 shows a multiplicative interaction in which the direction of influence of one variable depends on the magnitude of the other variable. Notice that when $x_2 = 0$, then an increase in the x_1 variable leads to a decline in y. But when $x_2 = 10$, then an increase in the x_1 variable leads to an increase in y. Again, the graph of the interaction shows a twist and is not flat.

A non-additive interaction of predictors does not have to be multiplicative. Other types of interaction are possible. The type of interaction is motivated by idiosyncratic theories in different variables in different application domains. Consider, for example, predicting the magnitude of gravitational force between two objects, from three predictor variables: mass of object one, mass of object two, and the distance between the objects. The force is proportional to the product (i.e., multiplication) of the two masses. But the force is proportional to the masses divided by the squared distance between them.[1]

14.1.6 Nominal predictors

14.1.6.1 Linear model for a single nominal predictor

The previous sections assumed that the predictor was metric. But what if the predictor is nominal, such as political party affiliation or gender? What is the simplest generic model for a metric variable predicted from a nominal variable? Answer: The "natural" model has each value of the nominal predictor generate a particular deflection of y away from its baseline level. For example, consider predicting height from sex (male or female). We can consider the overall average height across both sexes as the baseline height. When the individual has the value "male", that adds an upward deflection to the predicted height. When the individual has the value "female", that adds a downward deflection to the predicted height.

Expressing that idea in mathematical notation can get a little tricky. First consider the nominal predictor. We can't represent it appropriately as a single scalar value, such as 1 through 5, because that would mean that level 1 is closer to level 2 than it is to level 5, which is not true of nominal values. Therefore, instead of representing the value of the nominal predictor by a single scalar value, x, we will represent the nominal predictor by a vector $\vec{x} = \langle x_1, \ldots, x_J \rangle$, where J is the number of categories that the predictor has. When an individual has level j of the nominal predictor, this is represented by setting $x_j = 1$ and $x_{i,\,j} = 0$. For example, suppose x is sex, with level 1 being "male" and level 2 being female (so $J = 2$). Then "male" is represented as $\vec{x} = \langle 1, 0 \rangle$ and "female" is represented as $\vec{x} = \langle 0, 1 \rangle$. As another example, suppose that the predictor is political party affiliation, with Green as level 1, Democrat as level 2, Republican as level 3, Libertarian as level 4, and Other as level 5. Then Democrat is represented as $\vec{x} = \langle 0, 1, 0, 0, 0 \rangle$, and Libertarian is represented as $\vec{x} = \langle 0, 0, 0, 1, 0 \rangle$. Political party affiliation is being treated here as a categorical label only, with no ordering along a liberal-conservative scale.

Now that we have a formal representation for the nominal predictor variable, we can create a formal representation for the generic model of how the predictor influences the predicted variable. As mentioned above, the idea is that there is a baseline level of the predicted variable, and each category of the predictor indicates a deflection above or below that baseline level. We will denote the baseline value of the prediction as β_0. The deflection

[1] Division by squared distance can be thought of as multiplication by the reciprocal of squared distance, but that amounts to re-parameterizing the model.

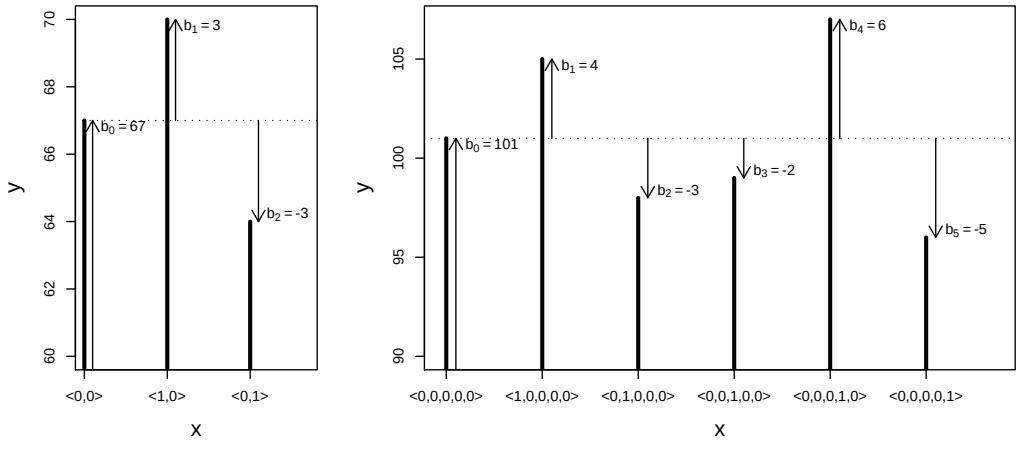

Figure 14.4: Examples of a nominal predictor (Equations 14.5 and 14.6). Left panel shows a case with $J = 2$, right panel shows a case with $J = 5$. In each panel, the baseline value of y is on the far left, when all the x components are zero. Notice that the deviations from baseline sum to zero.

for the j^{th} level of the predictor is denoted β_j. Then the predicted value is

$$\begin{aligned} y &= \beta_0 + \beta_1 x_1 + \cdots + \beta_J x_J \\ &= \beta_0 + \boldsymbol{\beta} \cdot \mathbf{x} \end{aligned} \tag{14.5}$$

where the notation $\boldsymbol{\beta} \cdot \mathbf{x}$ is sometimes called the "dot product" of the vectors.

Notice that Equation 14.5 has a form very similar to the basic linear form of Equation 14.1. The conceptual analogy is this: In Equation 14.1 for a metric predictor, the slope β_1 indicates how much y changes when x changes from 0 to 1. In Equation 14.5 for a nominal predictor, the coefficient β_j indicates how much y changes when x changes from neutral to category j.

There is one more consideration when expressing the influence of a nominal predictor as in Equation 14.5: How should the baseline value be set? Consider, for example, predicting height from sex. We could set the baseline height to be zero. Then the deviation from baseline for male might be 5'10" (say), and the deviation from baseline for female might be 5'4" (say). On the other hand, we could set the baseline height to be 5'7". Then the deviation from baseline for male would be "+3", and the deviation from baseline for female would be "−3". The second way of setting the baseline is the typical way it is done in generic statistical modeling. In other words, the baseline is constrained so that the deviations sum to zero across the categories:

$$\sum_{j=1}^{J} \beta_j = 0 \tag{14.6}$$

The expression of the model in Equation 14.5 is not complete without the constraint in 14.6.

Figure 14.4 shows examples of a nominal predictor, expressed in terms of Equations 14.5 and 14.6). The left panel shows a case for which $J = 2$, and the right panel

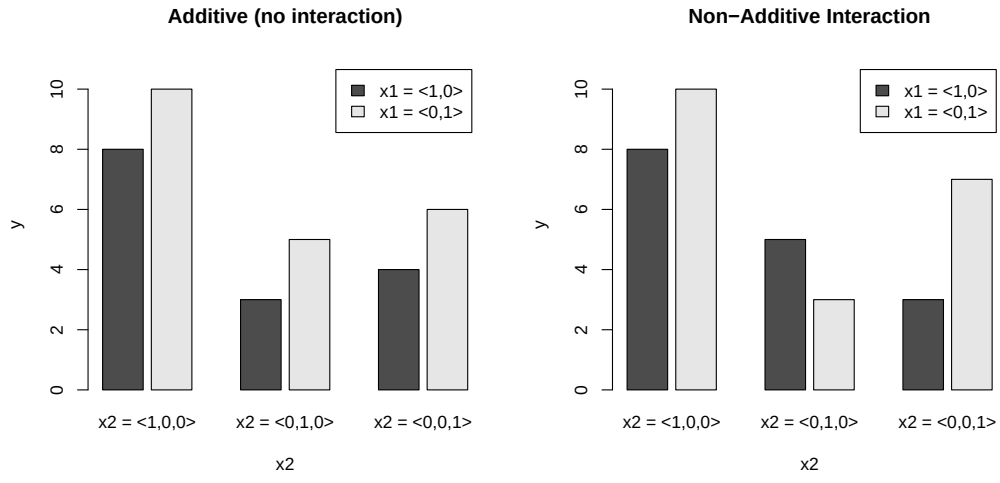

Figure 14.5: Combinations of two nominal variables. Left: Additive combination. Notice that the difference between adjacent dark-grey and light-grey bars is the same for every level of x_2. Right: Non-additive interaction. Notice that the difference between adjacent dark-grey and light-grey bars is the same for every level of x_2. Figure 19.1, p. 423, provides additional perspective and insight.

shows a case in which $\mu = 5$. Notice that the deflections from baseline sum to zero, as demanded by the constraint in Equation 14.6.

14.1.6.2 Additive combination of nominal predictors

Suppose we have two (or more) nominal predictors of a metric variable. For example, we might be interested in predicting income as a function of political party affiliation and gender. Figure 14.4 showed examples of each of those predictors individually. What we do now is consider the joint influence of those predictors. If the two influences are merely additive, then the model from Equation 14.5 becomes

$$\begin{aligned} y &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 \\ &= \beta_0 + \sum_{j=1}^{J_1} \beta_{1[j]} x_{1[j]} + \sum_{j=1}^{J_2} \beta_{2[j]} x_{2[j]} \end{aligned} \tag{14.7}$$

with the constraints

$$\sum_{j=1}^{J_1} \beta_{1[j]} = 0 \quad \text{and} \quad \sum_{j=1}^{J_2} \beta_{2[j]} = 0 \tag{14.8}$$

The left panel of Figure 14.5 shows an example of two nominal predictors that have additive effects on the predicted variable. In this case, the overall baseline is $\mu = 6$. When $x_1 = \langle 1,0 \rangle$, there is a deflection in y of -1, and when $x_1 = \langle 0,1 \rangle$, there is a deflection in y of $+1$. This deflection by x_1 is the same at every level of x_2. The deflections for the three levels of x_2 are $+3$, -2, and -1. These deflections are the same at all levels of x_1. Formally, the left panel of Figure 14.5 is expressed mathematically by the additive combination:

$$y = 6 + \langle -1, 1 \rangle x_1 + \langle 3, -2, -1 \rangle x_2$$

14.1.6.3 Nonadditive interaction of nominal predictors

When the predictor variables are non-metric, it does not even make sense to talk about a multiplicative interaction, because there are no numerical values to multiply. For example, consider predicting annual income from political party affiliation and gender. Both predictors are nominal, so it makes no sense to "multiply" them. But it does make sense to consider non-additive combination of their influences.[2]

For example, the overall influence of gender is that men, on average, have a higher income than women. The overall influence of political party affiliation is that Republicans, on average, have higher income than Democrats. But it may be that the influences combine non-additively: Perhaps people who are both Republican and male have a higher average income than would be predicted by merely adding the average income boosts for being Republican and for being male. (This interaction is not claimed to be true; it is being used only as a hypothetical example.)

We need new notation to formalize the non-additive influence of a combination of nominal values. Just as x_1 refers to the value of predictor 1, and x_2 refers to the value of predictor 2, the notation $x_{1,2}$ will refer to a particular *combination* of values of predictors 1 and 2. If there are J_1 levels of predictor 1 and J_2 levels of predictor 2, then there are $J_1 \cdot J_2$ combinations of the two predictors.

A non-additive interaction of predictors is formally represented by including a term for the influence of combinations of predictors, beyond the additive influences, as follows: $y = \beta_0 + \boldsymbol{\beta}_1 \mathbf{x}_1 + \boldsymbol{\beta}_2 \mathbf{x}_2 + \boldsymbol{\beta}_{1,2} \mathbf{x}_{1,2}$. Whenever the interaction coefficient $\boldsymbol{\beta}_{1,2}$ is non-zero, the predicted value of y is not a mere addition of the separate influences of the predictors.

The right panel of Figure 14.5 shows a graphical example of two nominal predictors that have interactive (i.e., non-additive) effects on the predicted variable. Notice, in the left pair of bars ($x_2 = \langle 1,0,0 \rangle$), that a change from $x_1 = \langle 1,0 \rangle$ to $x_1 = \langle 0,1 \rangle$ produces an increase of +2 in y, from $y = 8$ to $y = 10$. But for the middle pair of bars ($x_2 = \langle 0,1,0 \rangle$), a change from $x_1 = \langle 1,0 \rangle$ to $x_1 = \langle 0,1 \rangle$ produces an increase of 2 in y, from $y = 5$ to $y = 3$. Thus, the influence of x_1 is not the same at all levels of x_2.

An interesting aspect of the pattern in the right panel of Fig. 14.5 is that the average influences of x_1 and x_2 are the same as in the left panel. Overall, on average, going from $x_1 = \langle 1,0 \rangle$ to $x_1 = \langle 0,1 \rangle$ produces a change of 2 in y, in both the left and right panels. And overall, on average, for both panels it is the case that $x_2 = \langle 1,0,0 \rangle$ is +3 above baseline, $x_2 = \langle 0,1,0 \rangle$ is 2 below baseline, and $x_2 = \langle 0,0,1 \rangle$ is 1 below baseline. The only difference between the two panels is that the combined influence of the two predictors equals the sum of the individual influences in the left panel, but the combined influence of the two predictors does not equal the sum of the individual influences in the right panel.

An interaction between nominal predictors consists of any distinctive deflection, for each specific combination of categorical values, away from the additive combination. The magnitude of the interactive deflection is whatever is left over after the additive effects have been applied to the baseline. The model that includes an interaction term can be written as

$$y = \beta_0 + \boldsymbol{\beta}_1 \mathbf{x}_1 + \boldsymbol{\beta}_2 \mathbf{x}_2 + \boldsymbol{\beta}_{1,2} \mathbf{x}_{1,2} \tag{14.9}$$

[2] Actually, there is a sense in which nominal predictors can be multiplied to generate a predictor for interaction. Instead of ordinary multiplication of two scalars, we use the outer product of two vectors. The outer product was defined in Section 8.8.1, p. 144.

$$= \beta_0 + \sum_{j=1}^{J_1} \beta_{1;j} x_{1;j} + \sum_{k=1}^{J_2} \beta_{2;k} x_{2;k} + \sum_{j=1}^{J_1} \sum_{k=1}^{J_2} \beta_{1\ 2;j;k} x_{1\ 2;j;k}$$

with the constraints

$$\sum_{j=1}^{J_1} \beta_{1;j} = 0 \quad \text{and} \quad \sum_{k=1}^{J_2} \beta_{2;k} = 0 \quad \text{and} \quad \sum_{j=1}^{J_1} \beta_{1\ 2;j;k} = 0 \ 8k \quad \text{and} \quad \sum_{k=1}^{J_2} \beta_{1\ 2;j;k} = 0 \ 8j \quad (14.10)$$

In these equations, the term $x_{1\ 2}$ has J_1 times J_2 components, all of which are zero except for a 1 at the particular combination of levels of x_1 and x_2. This mysterious and arcane notation will be revealed in all its majestic grandeur in Chapter 19. For now, the main point is to understand that the term "interaction" refers to a non-additive influence of the predictors on the predicted, regardless of whether the predictors are measured on a nominal scale or a metric scale.

14.1.7 Linking combined predictors to the predicted

Once the predictor variables are combined, they need to be mapped to the predicted variable. This mathematical mapping is called the (inverse) link function, denoted by $f()$ in the following equation:

$$y = f(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{1\ 2} x_{1\ 2}) \quad (14.11)$$

Until now, we have been assuming that the link function is the identity function, $f(x) = x$. For example, in Equation 14.9, μ equals the linear combination of the predictors; there is no transformation of the linear combination before mapping the result to y.

Before describing different link functions, it is important to make some clarifications of terminology and corresponding concepts. First, the function $f()$ in Equation 14.11 is usually called the *inverse* link function, because the link function itself is thought of as transforming the value y into a form that can be linked to the linear model. I will abuse convention and simply refer to either $f()$ or $f^{-1}()$ as "the" link function, and rely on context to disambiguate which direction of linkage is intended. The reason for this terminological sadism is that the arrows in hierarchical diagrams of Bayesian models will flow from the linear model toward the data, and therefore it is natural for the functions to map toward the data, as in Equation 14.11. But repeatedly referring to this function as the "inverse" link would strain my patience and violate my aesthetic sensibilities. Second, the value μ that results from the link function $f(x)$ is not a data value per se. Instead, $f(x)$ is the value of a parameter that expresses some characteristic of the data, usually their mean. Therefore the function $f()$ in Equation 14.11 is sometimes called the mean function, and is written $\mu = f()$ instead of $y = f()$. I will not use this terminology because most students already think that "mean" means something else, namely the sum divided by N. The fact that μ in Equation 14.11 is a parameter value and not a data value will become clear in subsequent sections as we encounter specific cases and examples.

There are situations in which a non-identity link function is appropriate. Consider, for example, predicting response time as a function of amount of caffeine consumed. Response time declines as caffeine dosage increases, and therefore a linear prediction of RT from dosage would have a negative slope. This negative slope implies that for a very large dosage of caffeine, response time would become negative, which is impossible unless caffeine causes precognition (i.e., foreseeing events before they occur). Therefore a direct

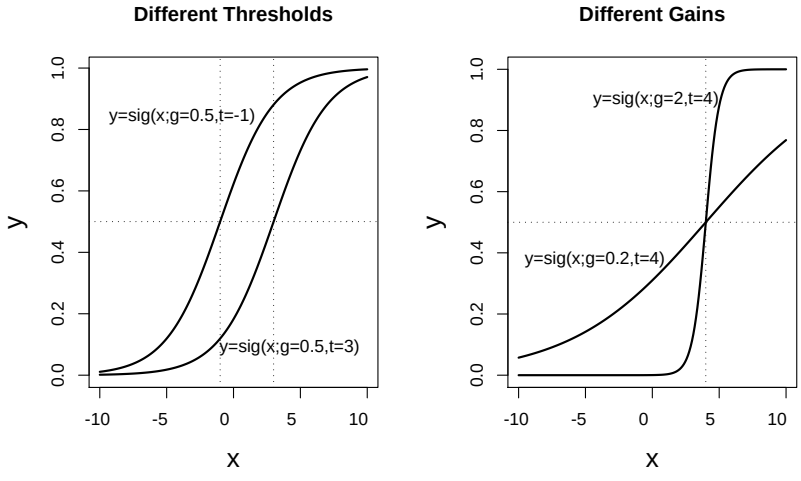

Figure 14.6: Examples of sigmoid functions of a single variable. The left panel shows sigmoids with the same gain but different thresholds. The right panel shows sigmoids with the same threshold but different gains.

linear function cannot be used for extrapolation to large scales, and we might instead want to use an exponential link function such as $y = \exp(\beta_0 + \beta_1 x)$.

14.1.7.1 The sigmoid (a.k.a. logistic) function

A frequently used link function is the sigmoid, also known as the logistic:

$$y = \mathrm{sig}(x) = 1 / (1 + \exp(-x)) \tag{14.12}$$

Notice the negative sign in front of the x. The sigmoid function ranges between 0 and 1. The sigmoid is nearly 0 when x is large negative, and is nearly 1 when x is large positive.

For linear combinations of predictors, the sigmoid link function is most conveniently parameterized in a threshold form. For a single predictor variable, the sigmoid link function applied to the linear function of the predictor yields

$$y = \mathrm{sig}(x; \gamma, \theta) = 1 / (1 + \exp(-\gamma(x - \theta))) \tag{14.13}$$

where γ, called the gain, corresponds to β_1 in Equation 14.2, and where θ, called the threshold, corresponds to $\beta_0 = -\beta_1\theta$ in Equation 14.2.

Examples of Equation 14.13, i.e., the sigmoid of a single predictor, are shown in Figure 14.6. Notice that the threshold is the point on the x axis for which $y = 0.5$. The gain indicates how steeply the sigmoid rises through that point.

Figure 14.7 shows examples of a sigmoid of two predictor variables. Above each panel is the equation for the corresponding graph. The equation is parameterized in a threshold form, as in Equation 14.4. In other words, $y = \mathrm{sig}(\sum_k w_k x_k)$, with $(\sum_k w_k^2)^{1/2} = 1$. Notice, in particular, that the coefficients of x_1 and x_2 in the plotted equations do indeed have Euclidean length of 1.0. For example, in the upper-right panel, $(0.71^2 + 0.71^2)^{1/2} = 1.0$, except for rounding error.

The coefficients of the x variables determine the orientation of the sigmoidal "cliff". For example, compare the two top panels in Figure 14.7, which differ only in the coefficients,

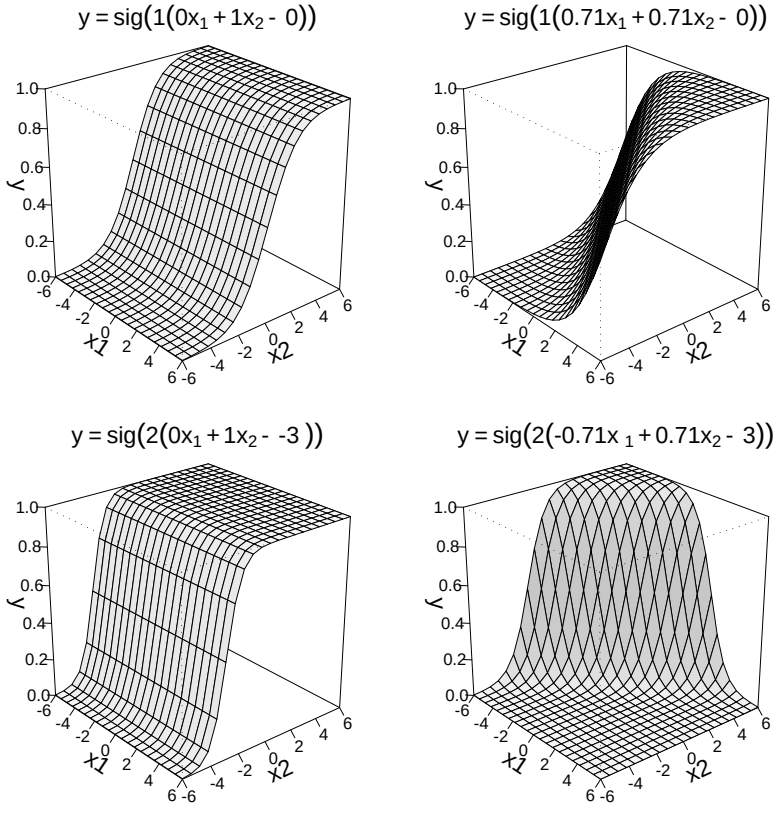

Figure 14.7: Examples of sigmoid functions of two variables. Top two panels show sigmoids with the same gain and threshold, but different coefficients on the predictors. The left two panels show sigmoids with the same coefficients on the predictors, but different gains and thresholds. The lower right panel shows a case with a negative coefficient on the first predictor.

not in gain or threshold. In the top left panel, the coefficients are $w_1 = 0$ and $w_2 = 1$, and the cliff rises in the x_2 direction. In the top right panel, the coefficients are $w_1 = 0.71$ and $w_2 = 0.71$, and the cliff rises in the positive diagonal direction.

The threshold determines the position of the sigmoidal cliff. In other words, the threshold determines the x values for which $y = 0.5$. For example, compare the two left panels of Figure 14.7. The coefficients are the same, but the thresholds (and gains) are different. In the upper left panel, the threshold is zero, and therefore the mid-level of the cliff is over $x_2 = 0$. In the lower left panel, the threshold is -3, and therefore the mid-level of the cliff is over $x_2 = -3$.

The gain determines the steepness of the sigmoidal cliff. Again compare the two left panels of Figure 14.7. The gain of the upper left is 1, whereas the gain of the lower left is 2.

Terminology: The logit function. The inverse of the logistic function is called the logit function. For $0 < p < 1$, $\text{logit}(p) = \log(p/(1-p))$. It is easy to show (try it!) that $\text{logit}(\text{sig}(x)) = x$, which is to say that the logit is indeed the inverse of the sigmoid. Some authors, and programmers, prefer to express the connection between predictors and predicted in the opposite direction, by first transforming the predicted variable to match the

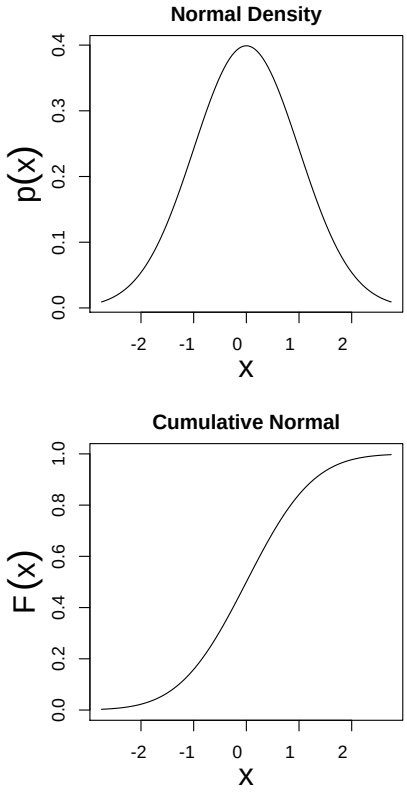

Figure 14.8: Top: A normal density with mean zero and precision one. Bottom: The corresponding cumulative normal function.

linear model. In other words, you may see the link expressed in one of these ways:

$$y = \text{logistic}(\beta_0 + \beta_1 x_1 + \ldots)$$
$$\text{logit}(y) = \beta_0 + \beta_1 x_1 + \ldots$$

The two expressions achieve the same result, mathematically. The difference between them is merely a matter of emphasis. In the first expression, the combination of predictors is transformed so it maps onto y, expressed in its original scale. In the second expression, y is transformed onto a new scale, and that transformed value is modeled as a combination of predictors.

14.1.7.2 The cumulative normal (a.k.a. Phi) function

Another frequently used link function is the cumulative normal distribution. It is qualitatively very similar to the sigmoid or logistic function. Modelers will use the logistic or the cumulative normal depending on mathematical convenience or ease of interpretation. For example, when we consider ordinal predicted variables (Chapter 21), it will be natural to model the responses in terms of a continuous underlying variable that has normally distributed variability, which leads to using the cumulative normal as a model of response probabilities.

The cumulative normal is denoted $\Phi(x; \mu, \tau)$, where x is a real number and where μ and τ are parameter values, called the mean and precision of the normal distribution. The parameter μ governs the point at which the cumulative normal, $\Phi(x; \mu, \tau)$, equals 0.5. In other words, μ plays the same role as the threshold in the logistic sigmoid. The parameter τ governs the steepness of the cumulative normal function near $x = \mu$. The τ parameter plays

the same role as the gain parameter in the logistic sigmoid. A graph of a cumulative normal appears in Figure 14.8. For this example, = 0, and notice that (0) = 0:5. This means that the area under the normal density to the left of 0 is 0.5.

Terminology: The probit function. The inverse of the cumulative normal is called the probit function. ("Probit" stands for "probability unit"; Bliss, 1934). The probit function maps a value p, for 0:0 p 1:0, onto the in nite real line, and a graph of the probit function looks very much like the logit function. You may see the link expressed either of these ways:

$$y = \quad _0 + \ _1 x_1 + : : :$$
$$\text{probit}(y) = \ _0 + \ _1 x_1 + : : :$$

Traditionally, the transformation of y (in this case, the probit function) is called the link function, and the transformation of the linear combination of x (in this case, the function) is called the inverse link function. As mentioned before, I use the traditional terminology and call either one a link function, relying on context to disambiguate.

14.1.8 Probabilistic prediction

In the real world, there is always variation in y that we cannot predict from x. This unpredictable "noise" in y might be deterministically caused by sundry factors we have neither measured nor controlled, or the noise might be caused by intrinsic non-determinism in y. It does not matter either way because in practice the best we can do is predict the probability that y will have any particular value, dependent upon x. Therefore we use the deterministic value predicted by Equation 14.11 as the predicted tendency of y as a function of the predictors. We do not predict that y is exactly $f\ _0 + \ _1 x_1 + \ _2 x_2 + \ _{1\ 2} x_{1\ 2}$ because we would surely be wrong. Instead, we predict that y tends to be near $f\ _0 + \ _1 x_1 + \ _2 x_2 + \ _{1\ 2} x_{1\ 2}$.

To make this notion of probabilistic tendency precise, we need to specify a probability distribution for y that depends on $f\ _0 + \ _1 x_1 + \ _2 x_2 + \ _{1\ 2} x_{1\ 2}$. To keep the notation tractable, first de ne $= f\ _0 + \ _1 x_1 + \ _2 x_2 + \ _{1\ 2} x_{1\ 2}$. Do not confuse this use of with the unrelated mentioned in the cumulative normal function. With this notation, we then denote the probability distribution of y as some to-be-speci ed probability density function, abbreviated as "pdf":

$$y \quad \text{pdf}(\quad [; \ ; :::])$$

The pdf might have various additional parameters, denoted by , to specify its shape. Examples are provided in the next section, where all these ideas are brought together.

14.1.9 Formal expression of the GLM

In general, the likelihood function specifies the probability of each possible predicted value of y as a function of the predictor values, x, and various parameter values, , etc. The generalized linear model can be written:

$$= f\ _0 + \ _1 x_1 + \ _2 x_2 + \ _{1\ 2} x_{1\ 2} + : : : \tag{14.14}$$
$$y \quad \text{pdf}(\quad [; \ ; :::]) \tag{14.15}$$

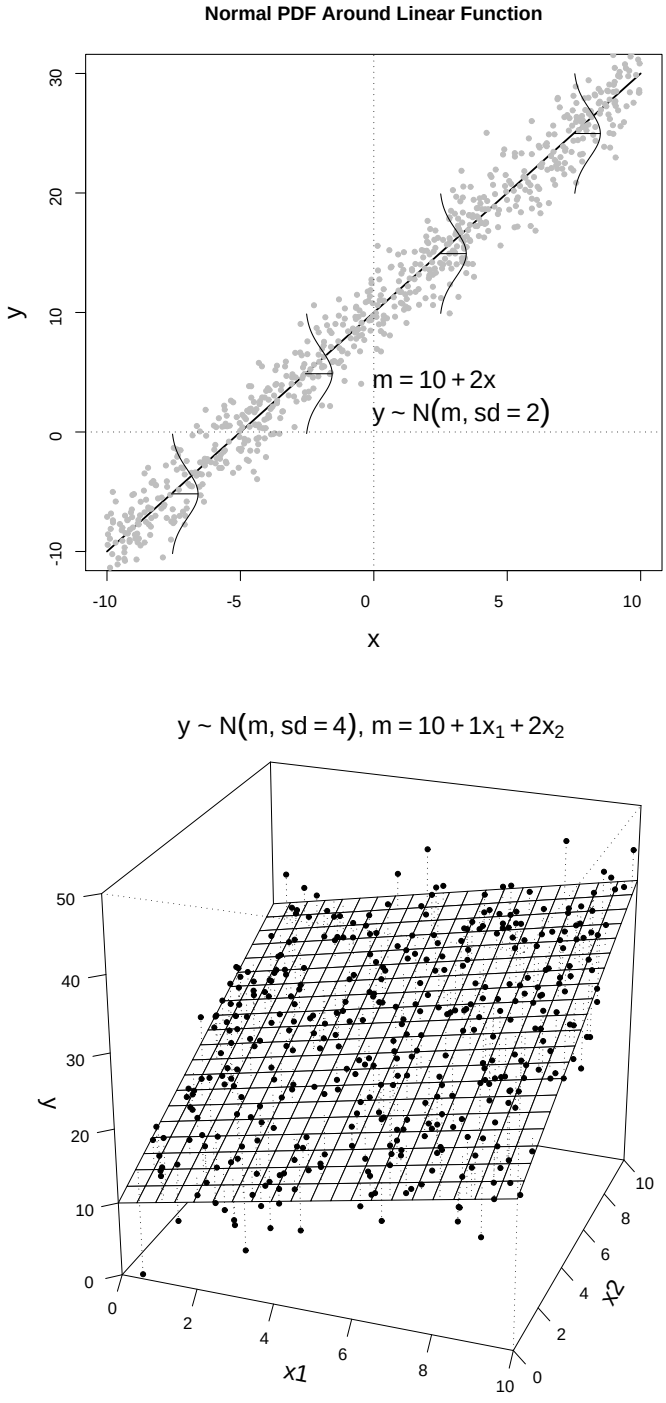

Figure 14.9: Examples of points normally distributed around a linear function.

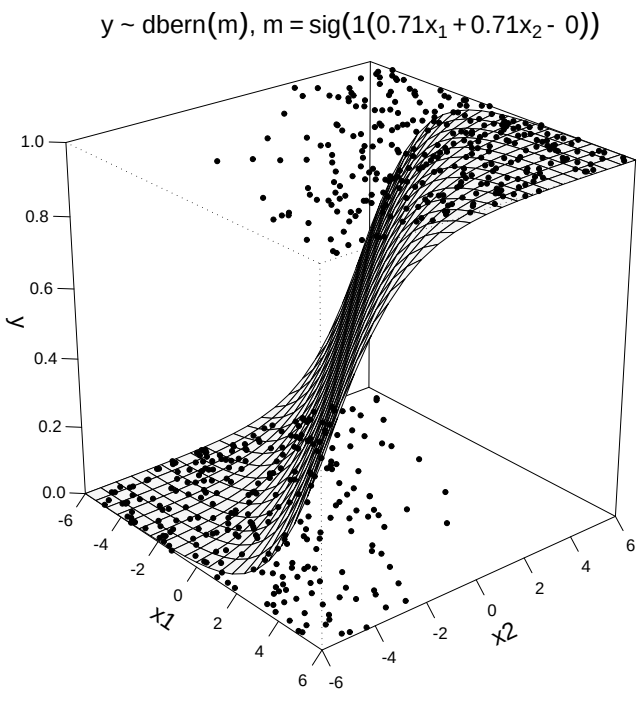

Figure 14.10: Examples of points Bernoulli distributed around a sigmoid function of two predictors. All the points are either at $y = 1$ or $y = 0$ (intermediate values such as $y = .6$ cannot occur).

The function f in Equation 14.14 is called the "link" function, because it links the combination of predictors to the predicted tendency. The optional parameters, $[\gamma, \ldots]$ in Equation 14.15 may be needed for various types of the probability density function (pdf) that describe the probability distribution of y around the tendency.

Figure 14.9 shows a random sample of points normally distributed around a line or plane. The upper panel illustrates a case of the generalized linear model of Equations 14.14 and 14.15 in which there is a single predictor, x, with $\beta_0 = 10$ and $\beta_1 = 2$. The link function is simply the identity function, $f(\beta_0 + \beta_1 x) = \beta_0 + \beta_1 x$. The probability density function is normal with a standard deviation of 2.0. Profiles of this normal density are superimposed on the graph to make it explicit. Notice that the normal density is always centered on the line that marks the predicted tendency as a function of the predictor.

The lower panel of Figure 14.9 shows a case with two predictor variables. The predictors are combined linearly, with no interaction. The link function is the identity. The probability function is normal with a standard deviation of 4. Each randomly generated point is connected to the underlying linear core by a vertical dotted line, to explicitly indicate the random variation of the point from the plane. The plane marks the predicted tendency as a function of the predictors, and the data values are normally distributed above and below that tendency.

Figure 14.10 shows another case of the GLM. In this case, the points are Bernoulli distributed around a sigmoid function of two predictors, as annotated at the top of the graph. There is a linear combination of predictors, with a sigmoid link function, and a Bernoulli probability function that defines the probability that $y = 1$. The graph shows that values of y can only be 0 or 1, and the sigmoid function defines the probability that y is 1 for particular

predictor values. The sigmoidal surface plots the tendency $y = 1$ as a function of the predictors.

14.2 Cases of the GLM

Table 14.1, p. 312, displays the various cases of the generalized linear model that are considered in this book. Subsequent chapters of the book progress through the table in reading order: left to right within rows, then top to bottom across rows.

The first row of Table 14.1 lists cases for which the predicted variable is metric. Moving from left to right within this row, the first column indicates a situation in which there is only a single group, and the predicted value for the group is simply the mean of the group. In this situation, there is no need to explicitly denote a predictor variable, and instead the mean of the group can be denoted by a single parameter, β_0. This situation corresponds to what in classical null hypothesis significance testing (NHST) is called the single-group t-test. This case is described in its Bayesian setting in Chapter 15.

Moving to the next column, there is a single metric predictor. This corresponds to so-called "simple linear regression", and is explored in Chapter 16. By inspecting the equation for the GLM in the cell, you can see that the only difference from the previous cell is the inclusion of the predictor x_1 and its coefficient β_1.

Moving rightward to the next column, we come to the scenario involving two or more metric predictors, which corresponds to "multiple regression," and is explored in Chapter 17. By examining the equations for the GLM in the cell, you can see that the basic form is the same, but merely with extra terms added for the additional predictors.

The next two columns involve nominal predictors, instead of metric predictors, with the penultimate column devoted to a single predictor and the last column devoted to two or more predictors. The last two columns correspond to what traditionally is called "oneway ANOVA" and "multifactor ANOVA". If that terminology is unfamiliar to you, don't worry, it will be explained in Chapters 18 and 19.

In all the cases in the first row, the link function is the identity function, and the probability distribution for the metric predicted values in assumed to be normal. When we move to the second row, however, the predicted variable is dichotomous, and therefore the probability distribution for y is a Bernoulli distribution. The link function, which connects the predictors to the probability that $y = 1$, is assumed to be the sigmoid, i.e., logistic function. When the predictors are metric, this situation is generically referred to as "logistic regression" and is discussed in Chapter 20. The case of nominal predictors is also discussed.

Finally, the bottom row of Table 14.1 lists cases for which the predicted variable is ordinal. These cases are considered in Chapter 21. Notice that the link function is the cumulative normal instead of the sigmoid, and the ordinal values are generated by multiple-category generalization of the Bernoulli function, denoted by dcat. Again, this will be explained at length in the forthcoming chapters. The point is for you to see the overall organization of topics, and to see how all these cases are variations of the same underlying structure.

The table can be expanded with additional rows and columns, but then it gets too big to display easily. Additional columns would include combinations of metric and nominal predictors. But it turns out that it is easy in Bayesian models to combine metric and nominal predictors, once you know how to handle metric and nominal predictors individually. Additional rows would involve different types of predicted variables. In particular, a fourth

Table 14.1: Cases of the generalized linear model (single predicted variable).

cted le y	Predictor Variable(s) x				
		Metric		Nominal	
	Single Group	Single x Factor	Two or More x Factors	Single x Factor	Two or More x Factors
ic	$= {}_0$ $y \quad N(\; ; \;)$	$= {}_0 + {}_1 x_1$ $y \quad N(\; ; \;)$	$= {}_0 + {}_1 x_1 + {}_2 x_2 + {}_{1\ 2} x_1 x_2$ $y \quad N(\; ; \;)$	$= {}_0 + {}_1 x_1$ $y \quad N(\; ; \;)$	$= {}_0 + {}_1 x_1 + {}_2 x_2 + {}_{1\ 2} x_{1\ 2}$ $y \quad N(\; ; \;)$
mous	$= {}_0$ $y \quad \text{dbern}( )$	$= \text{sig} \; {}_0 + {}_1 x_1$ $y \quad \text{dbern}( )$	$= \text{sig} \; {}_0 + {}_1 x_1 + {}_2 x_2 + {}_{1\ 2} x_1 x_2$ $y \quad \text{dbern}( )$	$= \text{sig} \; {}_0 + {}_1 x_1$ $y \quad \text{dbern}( )$	$= \text{sig} \; {}_0 + {}_1 x_1 + {}_2 x_2 + {}_{1\ 2} x_{1\ 2}$ $y \quad \text{dbern}( )$
nal		$= {}_0 + {}_1 x_1$ $_k = \text{thresh}_k( )$ $y \quad \text{dcat}(\ldots; \; {}_k; \ldots)$	$= {}_0 + {}_1 x_1 + {}_2 x_2 + {}_{1\ 2} x_1 x_2$ $_k = \text{thresh}_k( )$ $y \quad \text{dcat}(\ldots; \; {}_k; \ldots)$		

row would include count data for the predicted values. We will, in fact, cover one such case, as described in the next section. When the predicted data are count values, a natural link function is the exponential, and a natural pdf is the Poisson distribution, which will be de ned later in the book (Section 22.1.3). In summary, rows of the table refer to di erently scaled predicted values, with their corresponding link functions and pdf's:

y Scale Type	(Inverse) Link Function	pdf
metric	identity	normal
dichotomous	sigmoid (a.k.a. logistic)	Bernoulli
ordinal	thresholded cumulative normal	categorical
count	exponential	Poisson

14.2.1 Two or more nominal variables predicting frequency

Finally, we will also consider the situation in which there are two or more nominal variables used as predictors of a frequency count. A frequency count, i.e., how many times something happened, is a special case of a metric scale, but because values fall at discrete levels, namely non-negative integers, this situation will have a different sort of likelihood distribution. This type of situation, with nominal predictors and frequency-count predicted values, is often called "contingency table analysis" and a typical SPSS analysis conducts a "chi-square test of independence of attributes". We explore Bayesian analysis of this situation in Chapter 22.

Here is a brief summary of how contingency tables are analyzed using a model much like those in Table 14.1. In fact, a fourth row could be added to Table 14.1, with the predicted type labeled frequency count, and the model falling in the final column, under two nominal predictors. As a concrete example, suppose we record political a liation and religious a liation of a set of people, and for a sample of people we count how many occurrences there are of each combination. We are interested in analyzing possible relationships between political and religious a liations. Suppose we conduct a poll for one week. We happen to record 27 people who are Democrats and Unitarians. This observed frequency re ects an underlying rate at which that combination is generated by this sort of poll, i.e., the underlying rate for Unitarian Democrats is roughly 27 people per week. The observed rate (i.e., frequency per unit time) for each combination of nominal values is thought to re ect the true underlying rate at which that combination is generated by the world. We conceive of the observed rate as being a random sample from a true underlying rate denoted by . The probability of any particular observed rate, given an underlying rate of , is modeled by a Poisson distribution, which is denoted as dpois( ). The Poisson distribution was smuggled into the text back in Exercise 11.3, p. 235, which I'm sure is still as fresh in your memory as a beached sh. Don't worry, the Poisson will be explained again later (Section 22.1.3). The Poisson distribution specifies a probability for each possible observed rate. The Poisson puts highest probabilities on rates near .

Our goal is to estimate the underlying rates at which each nominal combination is produced. But more than that, we would like to know if the attributes occur independently of each other, or instead covary in some way. For example, if political and religious a liation are independent, then there should be the same proportion of Unitarians among Democrats as among Republicans. Mathematically, independence means that p(Unitarian&Democrat) = p(Unitarian) p(Democrat) and p(Unitarian&Republican) = p(Unitarian) p(Republican) and so on for every combination of attribute values. To smooth the expressions, I'll substi-

tute U for Unitarian and D for Democrat, whereby independence means

$$p(U \& D) = p(U) \quad p(D)$$

and so on for every combination of attributes. That expression for probabilities corresponds to the following expression in terms of frequencies:

$$freq(U \& D)=N = freq(U)=N \quad freq(D)=N$$

which can be re-arranged as

$$freq(U \& D) = freq(U) \quad freq(D) \quad 1=N:$$

Notice that independence is expressed as a multiplicative product of attribute influences. But all the models we've considered in this chapter used an additive sum of predictor influences. To be able to use our familiar additive models, we will transform the frequencies by a logarithm, because the logarithm of the product of values is the sum of the logarithms of the values. In other words,

$$\log(freq(U \& D)) = \log(freq(U)) + \log(freq(D)) + \log(1=N):$$

The notation "log(freq(value))" gets cumbersome, so we'll substitute the notation $_v$. Thus,

$$\log(freq(U \& D)) = \quad _U + \quad _D + \quad _0;$$

where $_0$ stands in for the constant $\log(1=N)$. Finally, it's unintuitive to talk about the logarithms of frequencies, so we'll exponentiate to get rid of the leading logarithm, yielding:

$$freq(U \& D) = \exp(\ _U + \ _D + \ _0):$$

To summarize, if independence is true, then the expression above should be true, for every combination of attribute values.

But of course the attributes are usually not independent, and we would like some measure of lack of independence. We already have such a measure in the context of linear models, namely, the interaction term. Thus, we will include an interaction term that estimates deviation from independence. Thus, our model ends up being as follows. For two nominal attributes, we put the observed frequencies in a table, with one attribute's values listed down the rows, and the other attribute's values listed across the columns. The frequency in the r^{th} row and c^{th} column is denoted $freq_{rc}$, and the underlying rate for that cell is denoted $_{rc}$. The model looks like this:

$$\begin{aligned} _{rc} &= \exp(\ _0 + \ _r + \ _c + \ _{rc}) \\ freq_{rc} &\quad dpois(\ _{rc}) \end{aligned} \tag{14.16}$$

with the usual constraints (from Equation 14.10)

$$\sum_r \ _r = 0 \quad \text{and} \quad \sum_c \ _c = 0 \quad \text{and} \quad \sum_r \ _{rc} = 0 \ 8c \quad \text{and} \quad \sum_c \ _{rc} = 0 \ 8r$$

The point of this over-fast prelude to contingency table analysis is merely to demonstrate that the core of the model we'll be using is the same linear model that was mentioned for multifactor ANOVA in Table 14.1. Thus, all the applied analyses we'll see in the remainder of the book are based on the GLM.

14.3 Exercises

Exercise 14.1. [Purpose: For real-world examples of research, identify which statistical model is relevant.] For each of the examples below, identify the predicted variable and its scale type, identify the predictor variable(s) and its scale type, and identify the cell of Table 14.1 to which the example belongs.

(A) Guber (1999) examined average performance by public high school students on the Scholastic Aptitude Test (SAT) as a function of how much money was spent per pupil by the state, and what percentage of eligible students actually took the exam.

(B) Hahn, Chater, and Richardson (2003) were interested in perceived similarity of simple geometric patterns. Human observers rated pairs of patterns for how similar the patterns appeared, by circling one of the digits 1–7 printed on the page, where 1 meant "very dissimilar" and 7 meant "very similar". The authors presented a theory of perceived similarity, in which patterns are perceived to be dissimilar to the extent that it takes more geometric transformations to produce one pattern from the other. The theory specified the exact number of transformations needed to get from one pattern to the other.

(C) R. L. Berger, Boos, and Guess (1988) were interested in the longevity of rats, measured in days, as a function of the rat's diet. One group of rats was fed freely, another group of rats had a very low calorie diet.

(D) McIntyre (1994) was interested in predicting the tar content of a cigarette (measured in milligrams) from the weight of the cigarette.

(E) You are interested in predicting the gender of a person, based on the person's height and weight.

(F) You are interested in predicting the whether a respondent will agree or disagree with the statement, "The United States needs a federal health plan with a public option", on the basis of the respondent's political party affiliation.

Exercise 14.2. [Purpose: Find student-relevant real-world examples of each type of situation in table 14.1.] For each of the twelve cells of Table 14.1 that is filled with two nominal variables, provide an example of research involving that cell's model structure. Do this by finding published articles that describe research with the corresponding structure. The articles do not need to have Bayesian data analysis; the articles do need to report research that involves the corresponding types of predictor and predicted variables. Because it might be overly time consuming to find published examples of all twelve types, just find published articles of at least six types. For each example, specify the following:

> The full citation to the published article (see the references of this book for examples of how to cite articles),
>
> The predictor and predicted variables. Describe their meaning and the type of scale they are. Briefly describe the meaningful context for the variables, i.e., what is the goal of the research.

Chapter 15

Metric Predicted Variable on a Single Group

Contents

> It's normal to want to fit in with your friends,
> Behave by their means and believe all their ends.
> But I'll be high tailing it, fast and askew,
> Precisely 'cause I can't abide what you do.

In this chapter, we consider a situation in which we have a single metric predicted variable that is observed for items from a single group. For example, we could measure the blood pressure (i.e., a metric variable) for people randomly sampled from first-year university students (i.e., a single group). In this case we might be interested in whether the group's typical blood pressure differs from the recommended value for people of that age as published by a federal agency. As another example, we could measure the IQ (i.e., a metric variable) of people randomly sampled from everyone self-described as vegetarian (i.e., a single group). In this case we could be interested in whether this group's IQ differs from the general population's average IQ of 100.

In the context of the generalized linear model (GLM) introduced in the previous chapter, this chapter's situation involves the most trivial case of the linear core of the GLM, namely

$= {}_0$, as indicated in the top-left cell of Table 14.1 (p. 312). The "news" of the present chapter is a detailed look at a particular probability density function, namely the normal distribution, denoted by $y \sim N(\ ;\ )$ in the top-left cell of Table 14.1. We will explore particular prior distributions for the parameters of the normal distribution, and see how those same density functions can be used in extended hierarchical models for research designs involving repeated measures. We will also provide examples of transforming data so they are approximately normal, so that "off the shelf" normal-likelihood models can be used to describe the data.

15.1 Estimating the mean and precision of a normal likelihood

The workhorse for the remainder of the book is the normal likelihood function, just as the Bernoulli likelihood was the focus for the previous part of the book. The normal probability density function was introduced in Section 3.3.2.2, p. 36. Please review that section now. The normal distribution specifies the probability density for a value y, given the values of two parameters, the mean and standard deviation:

$$p(y|\ ;\ ) = \frac{1}{Z}\exp\left(-\frac{1}{2}\frac{(y\quad)^2}{^2}\right) \tag{15.1}$$

where Z is the normalizer, i.e., a constant that makes the probability density integrate to 1. It turns out that $Z = \sqrt{2\ ^2}$, but we won't need to use this fact.

To get an intuition for the normal likelihood function, consider three data values $y_1 = 85, y_2 = 100$, and $y_3 = 115$, as plotted by large dots in Figure 15.1. The probability (density) of any single datum, given parameter values, is $p(y_i|\ ;\ )$ as specified in Equation 15.1. The probability of the whole set of independent data values is $p(y_i|\ ;\ ) = p(D|\ ;\ )$, where $D = \{y_1, y_2, y_3\}$. Figure 15.1 shows $p(D|\ ;\ )$ for different values of and . As you can see, there are values of and that make the data most probable, but other nearby values also accommodate the data reasonably well.

Given a set of data D, we estimate the parameters with Bayes' rule:

$$p(\ ;\ |D) = \frac{p(D|\ ;\ )p(\ ;\ )}{\int d\ \ d\ \ p(D|\ ;\ )p(\ ;\ )} \tag{15.2}$$

Figure 15.1 showed examples of $p(D|\ ;\ )$ for a particular data set at different values of and . The prior, $p(\ ;\ )$, specifies the believability of each combination of values in the two-dimensional conjoint parameter space, without the data. Bayes' rule says that the posterior believability of each combination of values is the prior believability times the likelihood, normalized by the evidence. We saw our first examples of Bayes' rule on a two-dimensional parameter space back in Chapter 8, for example in Figure 8.2, p. 134. Our goal now is to evaluate Equation 15.2, using a normal-density likelihood, for reasonable choices of the prior distribution, $p(\ ;\ )$.

15.1.1 Solution by mathematical analysis

Purely for the sake of mathematical elucidation, it is convenient to consider the case in which the standard deviation of the likelihood function is fixed at a specific value. In other words, the prior distribution on is a spike over that specific value. We'll denote the value

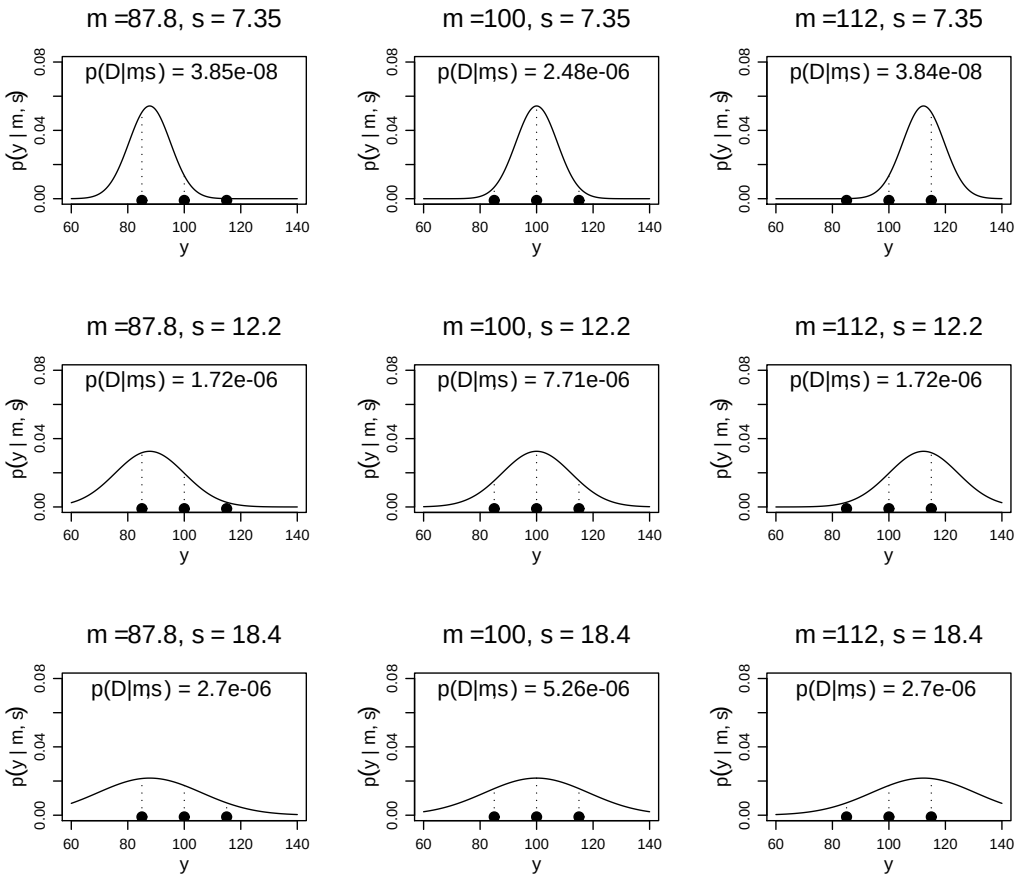

Figure 15.1: The likelihood $p(D|\mu, \sigma)$ for three data points $D = \{85, 100, 115\}$ according to a normal likelihood function with different values of μ and σ. The probability density of an individual datum is the height of the dotted line over the point. The probability of the set of data is the product of the individual probabilities. The middle panel shows shows the μ and σ that maximize the probability of the data.

as $\sigma = S_y$. With this simplifying assumption, we are only estimating μ because we are assuming perfectly certain prior knowledge about σ.

When σ is fixed, then the prior distribution on μ in Equation 15.2 can be easily chosen to be conjugate to the normal likelihood. (The term "conjugate prior" was defined in Section 5.2, p. 67). It turns out that the product of normal distributions is again a normal distribution; in other words, if the prior on μ is normal, then the posterior on μ is normal. It is easy to derive this fact, as we do next.

Let the prior distribution on μ be normal with mean M and standard deviation S. Then the likelihood times the prior is

$$p(y|\mu, \sigma)p(\mu, \sigma) = p(y|\mu, S_y)p(\mu)$$
$$\propto \exp\left(-\frac{1}{2}\frac{(y-\mu)^2}{S_y^2}\right)\exp\left(-\frac{1}{2}\frac{M-\mu}{S^2}\right)$$

$$
\begin{aligned}
&= \exp\left(-\frac{1}{2}\left[\frac{(y-\mu)^2}{S_y^2} + \frac{(\mu-M)^2}{S^2}\right]\right) \\
&= \exp\left(-\frac{1}{2}\left[\frac{S^2(y-\mu)^2 + S_y^2(\mu-M)^2}{S_y^2 S^2}\right]\right) \\
&= \exp\left(-\frac{1}{2}\left[\frac{S_y^2+S^2}{S_y^2S^2}\left(\mu^2 - 2\frac{S_y^2M_u + S^2y}{S_y^2+S^2}\mu + \frac{S_y^2M_u^2 + S^2y^2}{S_y^2+S^2}\right)\right]\right) \\
&= \exp\left(-\frac{1}{2}\left[\frac{S_y^2+S^2}{S_y^2S^2}\left(\mu^2 - 2\frac{S_y^2M_u + S^2y}{S_y^2+S^2}\mu\right)\right]\right) \\
&\quad \exp\left(-\frac{1}{2}\left[\frac{S_y^2+S^2}{S_y^2S^2}\left(\frac{S_y^2M_u^2 + S^2y^2}{S_y^2+S^2}\right)\right]\right) \\
&\propto \exp\left(-\frac{1}{2}\left[\frac{S_y^2+S^2}{S_y^2S^2}\left(\mu^2 - 2\frac{S_y^2M_u + S^2y}{S_y^2+S^2}\mu\right)\right]\right)
\end{aligned}
\tag{15.3}
$$

where the transition to the last line was valid because the term that was dropped was merely a constant. This result, believe it or not, is progress. Why? Because we ended up, in the innermost parentheses, with a quadratic expression in μ. Notice that the normal prior is also a quadratic expression in μ. All we have to do is "complete the square" inside the parentheses, and do the same trick that got us to the last line of Equation 15.3:

$$
\begin{aligned}
p(\mu|y, S_y)p(\mu) &\propto \exp\left(-\frac{1}{2}\left[\frac{S_y^2+S^2}{S_y^2S^2}\left(\mu^2 - 2\frac{S_y^2M_u + S^2y}{S_y^2+S^2}\mu\right) + \left(\frac{S_y^2M_u + S^2y}{S_y^2+S^2}\right)^2\right]\right) \\
&= \exp\left(-\frac{1}{2}\left[\frac{S_y^2+S^2}{S_y^2S^2}\left(\mu - \frac{S_y^2M_u + S^2y}{S_y^2+S^2}\right)^2\right]\right)
\end{aligned}
\tag{15.4}
$$

Equation 15.4 is the numerator of Bayes' rule. When it is normalized by the evidence in the denominator, it becomes a probability density function. What is the shape of the function? You can see that Equation 15.4 has the same form as a normal distribution on μ, such that the mean is $\frac{S_y^2M + S^2y}{S_y^2+S^2}$ and the standard deviation is $\sqrt{\frac{S_y^2S^2}{S_y^2+S^2}}$.

That formula is rather unwieldy! It becomes more compact if the normal density is re-expressed in terms of $\tau = 1/\sigma^2$ instead of σ. The reciprocal of the squared standard deviation is called the *precision* of the normal. When the standard deviation goes down, the precision goes up. A very narrow distribution is highly precise. A wide distribution has low precision. Because the posterior standard deviation is $\sqrt{\frac{S_y^2S^2}{S_y^2+S^2}}$, the posterior precision is

$$\frac{S_y^2 + S^2}{S_y^2S^2} = \frac{1}{S^2} + \frac{1}{S_y^2}.$$

In other words —and this is the punch line— the posterior precision is the sum of the prior precision and the likelihood precision.

The posterior mean can also be re-expressed in terms of precisions. The posterior mean

is $\frac{S_y^2 M + S^2 y}{S_y^2 + S^2}$, which becomes

$$\frac{1/S^2}{1/S_y^2 + 1/S^2} M + \frac{1/S_y^2}{1/S_y^2 + 1/S^2} y.$$

In other words, the posterior mean is a weighted average of the prior mean and the datum, with the weighting corresponding to the relative precision of the prior and the likelihood. When the prior is highly precise compared to the likelihood, i.e., when $1/S^2$ is large compared to $1/S_y^2$, then the prior is weighted heavily and the posterior mean is near the prior mean. But when the prior is imprecise, i.e., very uncertain, then the prior does not get much weight and the posterior mean is close to the datum. We have previously seen this sort of relative weighting of prior and data in the posterior. It showed up in the case of updating a beta prior, back in Equation 5.8, p. 71.

The formulas for the mean and precision of the posterior can be naturally extended when there are N values of y in a sample, instead of only a single value of y. The formulas can be derived from the defining formulas, as was done above, but a short cut can be taken. It is known from mathematical statistics that when a set of values y_i are generated from a normal likelihood function, the mean of those values, $\bar{y}$, is also distributed normally, with the same mean as the the generating mean, and with standard deviation of $= \sqrt{N}$. Thus, instead of conceiving of this situation as N scores y_i sampled from the likelihood $N(y_i|\ ;\ )$, we conceive of this as a single score, $\bar{y}$, sampled from the likelihood $N(\bar{y}|\ ;\ = \sqrt{N})$. Then we just apply the updating formulas we previously derived. Thus, for N scores y_i generated from a normal likelihood $N(y_i|\ ;\ S_y)$ and prior distribution $N(\ |M\ ;S\ )$ on , the posterior distribution on is also normal with mean

$$\frac{1/S^2}{N/S_y^2 + 1/S^2} M + \frac{N/S_y^2}{N/S_y^2 + 1/S^2} \bar{y}$$

and precision

$$\frac{1}{S^2} + \frac{N}{S_y^2}.$$

Notice that as the sample size N increases, the posterior mean is dominated by the data mean.

Instead of estimating the parameter in the likelihood when is xed, we can estimate the parameter when is xed. The situation is also more conveniently expressed in terms of precision: We want to estimate the precision $= 1/^2$ when is xed. It turns out that when is xed, a conjugate prior for the precision is the gamma distribution (e.g., Gelman et al., 2004, p. 50). The gamma distribution was described in Figure 9.8, p. 170. For our purposes, it is not important to review the updating formulas for the gamma distribution in this situation. But it is important to gain an intuition for what a gamma prior on precision means, in terms of the beliefs represented. Consider a gamma distribution that is loaded heavily over very small values, but has a long shallow tail leading over large values. This sort of gamma distribution on precision indicates that we believe most strongly in small precisions, but we admit that large precisions are possible. If this is a belief about the precision of a normal likelihood function, then this sort of gamma distribution expresses a belief that the data will be very spread out, because small precisions imply large standard deviations. If the gamma distribution is loaded over large values of precision, it expresses a belief that the data will be tightly clustered.

Summary. We have assumed that the data are generated by a normal likelihood function, parameterized by a mean and standard deviation , and denoted y N(yj ;). For purposes of mathematical derivation, we made unrealistic assumptions that the prior distribution is either a spike on or a spike on , in order to make three main points:

1. A natural way to express a prior on is with a normal distribution, because this is conjugate with the normal likelihood when its precision is xed.
2. A natural way to express a prior on the precision = 1 is with a gamma distribution, because this is conjugate with the normal likelihood when its mean is xed.
3. The formulas for Bayesian updating of the parameter distribution are more conveniently expressed in terms of precision than standard deviation. Normal distributions will be described sometimes in terms of standard deviation and sometimes in terms of precision, so it is important to glean from context which is being referred to.

A joint prior, on the combination of and parameter values, can also be speci ed, in such a way that the posterior has the same form as the prior. We will not pursue these mathematical analyses here, because our purpose is merely to justify and motivate typical expressions for the prior distributions on the parameters, so that they can then be used in MCMC sampling in BUGS. Various other sources describe conjugate priors for the joint parameter space (e.g., Gelman et al., 2004, pp. 78–83).

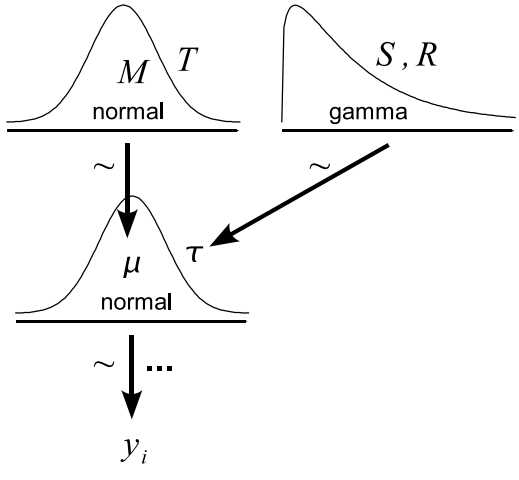

Figure 15.2: A model of dependencies for several metric scores drawn from a single group. The normal distributions are parameterized by mean and precision (not standard deviation).

15.1.2 Approximation by MCMC in BUGS

It is easy to estimate the mean and precision of a set of data in BUGS. Figure 15.2 illustrates the model. The data y_i, are assumed to be generated by a normal likelihood function with mean and precision . The prior on is assumed to be normal with mean M and precision T. The prior on is assumed to be gamma with shape and rate parameters of S and R. We are estimating two parameters, and the prior is over the two-dimensional conjoint parameter space. The prior we have speci ed assumes that and are independent.

The model of Figure 15.2 is expressed in BUGS as follows: (YmetricXsingleBrugs.R)

```
model {
    # Likelihood:
    for( i in 1 : N ) {
        y[i] ~ dnorm( mu , tau ) # tau is precision, not SD
    }
    # Prior:
    tau ~ dgamma( 0.01 , 0.01 )
    mu ~ dnorm( 0 , 1.0E-10 )
}
```

Notice that each arrow in Figure 15.2 has a corresponding specification in the BUGS code. Notice that BUGS parameterizes the *dnorm* by mean and precision, not by mean and standard deviation. The hyperprior constants in this particular BUGS model are generic and uninformed; they should be knowledgably specified in real applications.

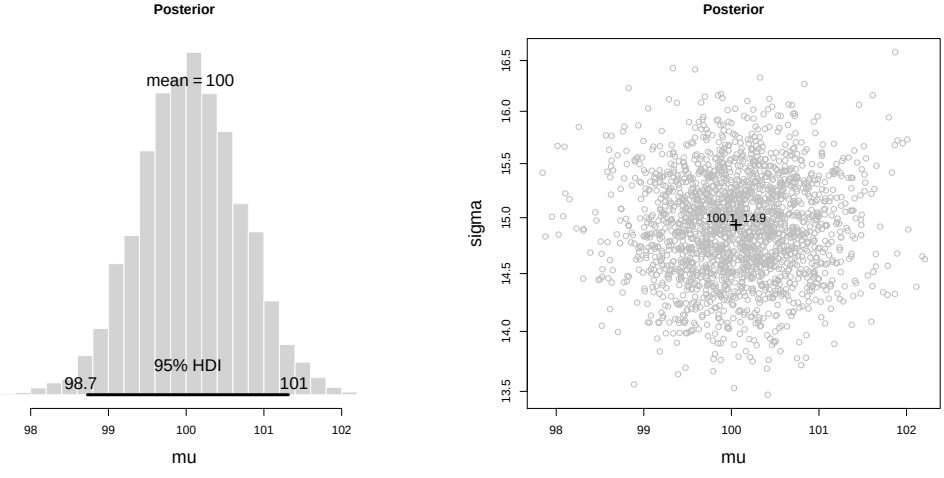

Figure 15.3: Posterior distribution from program in Section 15.4.1 (YmetricXsingleBrugs.R).

When learning about or debugging a model, it can be useful to generate fictitious data from known parameter values, and then see how well those parameter values are estimated by the model's posterior distribution. For this purpose, the program in Section 15.4.1 (YmetricXsingleBrugs.R) generates some random data from a normal distribution whose true mean is 100 and standard deviation is 15 (just like IQ scores). The random data themselves have a sample mean and standard deviation somewhat different from those generating values. The result of running the BRugs program is shown in Figure 15.3. The posterior precision (i.e., tau) sampled by the BUGS program has been converted to standard deviation (denoted "sigma") via the identity $= 1=\sqrt{}$. The estimated mean and standard deviation are not far from the generating mean. This reassures us that the program is operating properly.

15.1.3 Outliers and robust estimation: The t distribution

Figure 15.4 shows examples of the t distribution. The t distribution was originally invented by Gosset (1908), who used the pseudonym "Student" because his employer (Guinness

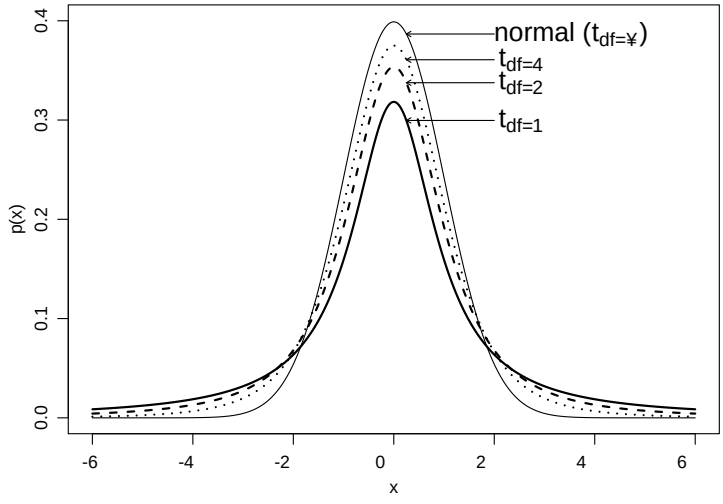

Figure 15.4: t distributions, with a normal distribution superimposed for comparison. The t distribution has tall tails relative to the normal. The df parameter ("degrees of freedom") controls the relative height of the tails.

Brewery) prohibited publication of any research that might reveal proprietary or imply problems with their product (such as variability in quality). Therefore the distribution is often referred to as the Student t distribution.

In R, the t density at x is specified by dt(x,df), where df is a parameter called the degrees of freedom. (The R function dt also has an optional argument called ncp, the noncentrality parameter. This option is not implemented in BUGS dt function.) The effect of the df argument is shown in the Figure 15.4. When df is small, the t distribution has tails that are taller than normal. As df approaches infinity, the t distribution becomes normal. The df parameter is a continuous value greater than or equal to one. (Readers might be familiar with the df parameter as related to sample size when t is a sampling distribution, but the t distribution is not used as a sampling distribution here, and therefore df is not restricted to being an integer.) In BUGS, the density is specified by dt(mu,tau,df). The mu and tau arguments in BUGS have no equivalent arguments in R. All they do is linearly shift the t distribution, such that dt(mu,tau,df) in BUGS corresponds to sqrt(tau) * dt((x-mu)*sqrt(tau) , df) in R.

Although the t distribution was originally conceived as a sampling distribution for NHST, we will use it instead as a convenient model of data with outliers (as is often done; e.g., Damgaard, 2007; Meyer & Yu, 2000; Tsionas, 2002). Outliers are simply data values that fall unusually far from the model's expected value. Real data often contain outliers, presumably because some extraneous influences have somehow perturbed the data values. Sometimes these extraneous influences can be identified and the affected data values can be corrected. But usually we have no way of knowing whether a suspected outlying value was caused by an extraneous influence, or is a genuine reflection of the target being measured. Instead of deleting suspected outliers from a data set according to some arbitrary criterion, we retain all the data but use a likelihood function that is less affected by outliers than is the normal distribution.

Figure 15.5 shows examples of how the t distribution is robust against outliers. The curves show the maximum likelihood estimates (MLEs) of the parameters for the t and

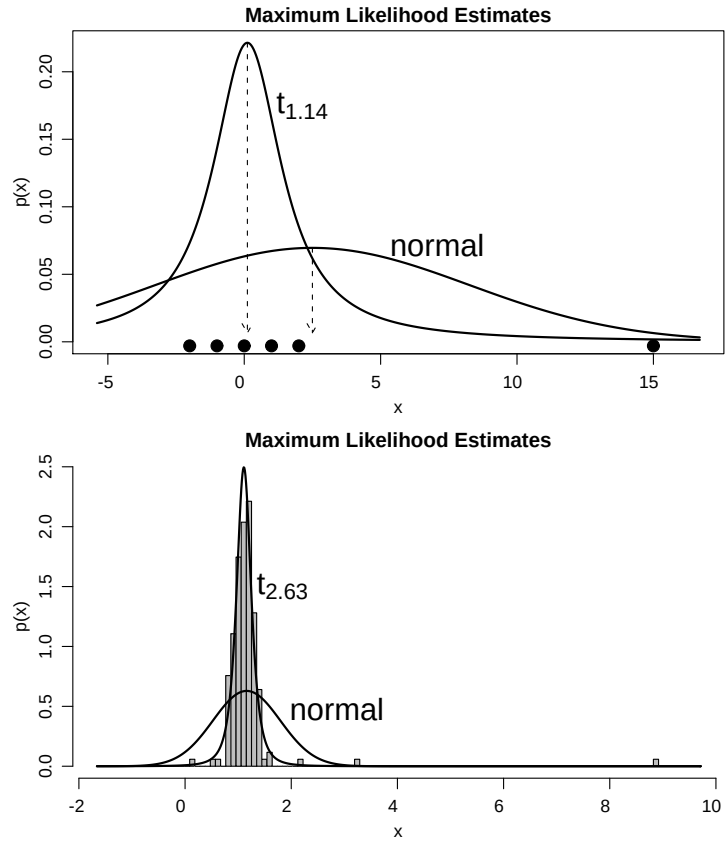

Figure 15.5: The maximum likelihood estimates of normal and t distributions to the data shown. Upper panel shows "toy" data to illustrate that the normal accommodates an outlier only by enlarging its standard deviation and, in this case, by shifting its mean. Lower panel shows actual data to illustrate realistic effect of outliers on estimates of the normal.

normal distributions. More formally, for the given data $D = \{x_i\}$ parameter values were found for the normal that maximized $p(D|\mu, \sigma)$, and parameter values were found for the t distribution that maximized $p(D|\mu, \sigma, df)$, and the curves of those MLEs are plotted with the data. The upper panel of Figure 15.5 shows "toy" data to illustrate that the normal is strongly influenced by an outlier, while the t distribution remains centered over the bulk of the data. The lower panel of Figure 15.5 uses realistic data that indicates levels of inorganic phosphorous, measured in milligrams per deciliter, of 177 human subjects aged 65 or older. The authors of the data (Holcomb & Spalsbury, 2005) intentionally altered a few data points to reflect typical transcription errors and to illustrate methods for detecting and correcting such errors. We instead assume that we never have access to records of the original individual measurements, and must model the uncorrected data set. The t distribution accommodates the outliers and fits the distribution of data much better than the normal.

The t distribution is useful as a likelihood function, for modeling outliers at the level of observed data, but it is also useful for modeling outliers at higher levels in a hierarchical prior. We will encounter such applications when we model individual differences or multiple predictors, e.g., in Section 16.2.

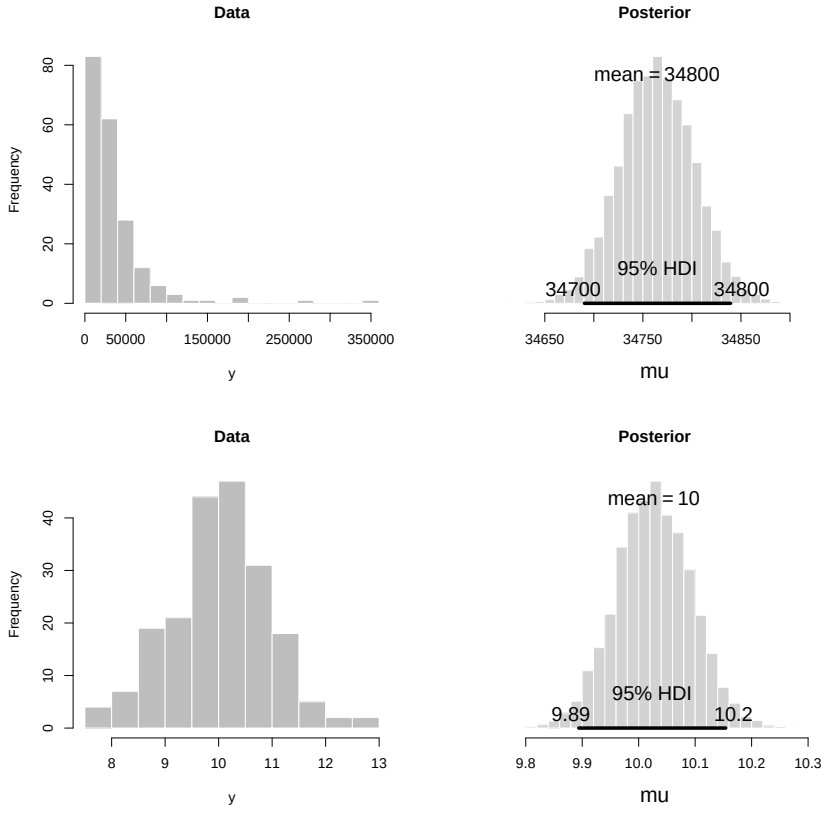

Figure 15.6: Upper left shows skewed, non-normal data. Upper right shows the posterior distribution of when the data are modeled (inappropriately!) with a normal likelihood. Bottom left shows the same data transformed, via a logarithm. Bottom right shows the posterior distribution of when the transformed data are modeled (appropriately) with a normal likelihood.

15.1.4 When the data are non-normal: Transformations

Suppose there has been discovered a new species of insects that live in dark caves, and the insects make ultra high frequency sound bursts. One theory is that the sounds should have an average frequency of 22,000 Hz (cycles per second). An entomologist has measured the frequencies of 200 sound bursts from these insects, and would like to know if a mean of 22,000 Hz is credible.

The upper left panel of Figure 15.6 shows a histogram of the data, and the upper right panel shows the posterior distribution of from the normal-likelihood model of Figure 15.3. (Assume that the prior was uncontroversial.) According to the posterior, the data exclude a value of 22,000. Therefore the entomologist concludes that the 22,000 Hz theory is not credible.

The proponent of the theory says that the theory was misstated. Instead of scaling the insect sounds by their raw frequency, they should be scaled by their perceived pitch, which is merely the logarithm of frequency. The natural log of 22,000 is 10. Thus, the theory claims that the sound bursts should have an average (log frequency) of 10. The lower left panel of Figure 15.6 shows the logarithm of the same data as the upper left panel. When the normal-likelihood model is applied to these data, the resulting posterior on

is shown in the lower right of Figure 15.6. (Assume that the prior was uncontroversial.) According to this posterior, the data do not exclude a value of 10. Indeed, the value of 10 is very near the mode of the posterior distribution.

Which analysis do we trust? Is the theory disconfirmed or upheld? In this case, the theory is clearly upheld, because the analysis based on raw frequency is not appropriate. The problem with the analysis based on raw frequency is that the normal-likelihood model assumes that the data are normally distributed, but the raw frequencies are obviously extremely skewed and non-normal. Graphically, the normal likelihood model in Figure 15.2 shows that the data, y, are generated by a normal distribution. The parameters μ and σ are only meaningful to the extent that the normal likelihood actually describes the distribution of the data. Because of the severe skew in the raw-frequency data, the central-tendency parameter μ is being pulled far to the right by the extreme values far above 2,000, for which there are no symmetrically distributed values below 2,000, as is assumed by a symmetric normal distribution. The logarithmically transformed data, on the other hand, are apparently much more normally distributed, and therefore the normal-likelihood model is a viable description of the data.

The moral of the story is this: When using a normal-likelihood model, the data should be at least roughly normally distributed, otherwise the model is a poor description of the data and the model's parameter values may be meaningless. When there is only moderate non-normality in the data, the misrepresentation by the model parameteres is often hardly noticeable, but when there is severe non-normality in the data, then the misrepresentation may be substantial.

There are many real-world situations in which data are not normally distributed. A prominent example is response times: Ask a person to press a button as quickly as possible after a light comes on. Repeat this many times. The person's response times will be strongly skewed toward higher values. This skew makes intuitive sense because the response time can't be much faster than one or two hundred milliseconds, but there are many reasons to be slow on some trials. Another example is shown in Figure 15.1, which reveals that the longevity of lab rats on a restricted diet is skewed to the left.

How should we analyze non-normal data? There are two possible solutions. One method perseveres with a normal-likelihood function, but transforms the data so that they are very nearly normal. There is nothing wrong with transforming data, as long as all the data are transformed the same way, and the transformation preserves order. A transformation that preserves order is called "monotonic". Examples of monotonic transformations include the exponential function, the logarithm (for positive-valued data), and the cubic (i.e., y^3). An example of a non-monotonic transformation is a sine wave function, when the original data span several cycles of the sine wave. Monotonic transformations merely re-scale the data. The measurement scale is arbitrary. Earthquake intensities could be measured in terms of raw energy, or in terms of the base-10 logarithm of energy, which is the Richter scale. Temperatures can be measured in terms of Fahrenheit or Celsius scales. Distances can be measured in terms of the English or Metric system. Acoustic vibrations can be measured by frequency or by perceived pitch, which is essentially the logarithm of frequency. The analysis should use whatever transformation makes the data normally distributed, so that the data respect the assumption of the normal-likelihood model and thereby make the parameter estimates meaningful.

The second approach to analyzing non-normal data is to stick with the normal likelihood function, and use a more appropriate likelihood function instead. For example, Rouder, Lu, Speckman, Sun, and Jiang (2005) used a Weibull distribution as the likelihood function

for modeling skewed response time distributions. This approach requires familiarity with various non-normal probability distributions. Moreover, for use in BUGS, this method also requires that the desired likelihood function can be specified in BUGS. Because of these extra complications, people usually first try the method of transforming data to normal. But a non-normal likelihood function may be preferred when no reasonable data transformation succeeds, or when transformed scales are too awkward to work with, or when a theory posits a specific non-normal distribution for generating data.

There is one more important caveat when transforming data (and changing likelihood functions): The prior must be appropriate to the transformed data. For example, when the insect sounds were measured in terms of raw frequency, the prior should be appropriate for data in the broad vicinity of 22,000. But when the insect sounds were measured by log(frequency), then the prior should be appropriate for data in the general vicinity of 10.

Exercise 15.2 provides a realistic example in which we are interested in estimating the underlying mean that generated a set of scores. It also asks us to consider the choice of prior, the robustness of the posterior, and whether a t likelihood seems to be an appropriate model of the data. Exercise 15.3 gives another realistic example in which the data are skewed.

The section previous to this one discussed outliers. Can outliers be "transformed away"? In simple, one-group data sets, the answer, in principle, is yes. We merely fashion a transformation that arbitrarily compresses the extreme tails of the distribution in just the right way. But then the prior should take into account the transformation. Unfortunately, for more typical complex situations, it is difficult to invent a transformation that appropriately compresses all outliers simultaneously. Therefore, transformations are typically used to adjust large-scale skew in the data, and not for outliers. Outliers, on the other hand, are typically addressed by using tall-tail distributions, such as the t distribution.

15.2 Repeated measures and individual differences

In many real-world applications, we have repeated measures from the same individual, although we are mainly interested in the central tendency of the group. For example, we might be interested in the typical blood pressure of employees at a company. We can repeatedly measure the blood pressure of each employee at random times of day and on random different days. Our overall goal is to estimate the typical blood pressure of the group as a whole, but now we have repeated measurements from individual subjects.

We assume that the individual subjects have been randomly sampled from the pool of all possible subjects in the population of interest (e.g., the pool of all employees in the company). We also assume that the repeated measurements within each subject are mutually independent. This is sometimes a perilous assumption because it is probably only approximately true even if we are careful to design the measurement procedure with the assumption in mind. For example, suppose we are measuring someone's blood pressure, with the goal of getting measurements that represent that person's typical blood pressure. We could measure the blood pressure several times in rapid succession, but presumably those measurements would be highly correlated because the person's blood pressure would not change much in only a few seconds duration. Instead, we should measure blood pressure at random times across hours and days. The goal is to make the repeated measurements far enough apart in time so that we can reasonably assume that the repeated measurements are independent of each other. But we don't want the repeated measurements to be so far apart

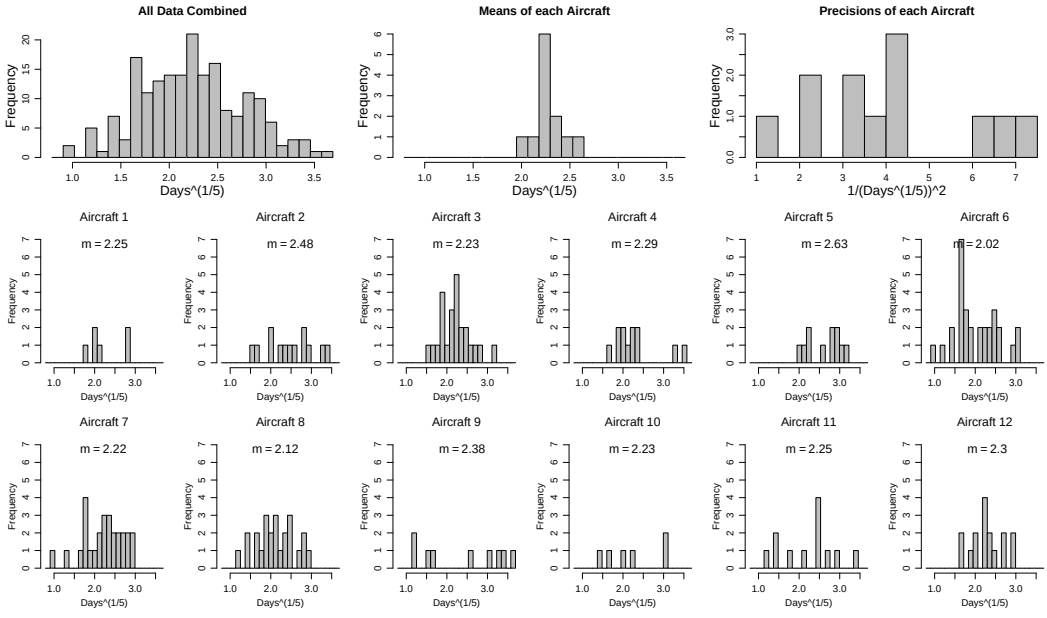

Figure 15.7: Number of days between failures of air-conditioning systems in aircraft. Notice the scale on the x-axis. The lower 12 panels show data from individual aircraft. The upper-left panel shows all those data combined into one histogram. The upper-middle panel shows a histogram of the 12 aircraft means.

in time that the person's underlying typical value has changed, e.g., from dramatic change in body fat. Thus, we want the measurements to be far enough apart that they have minimal correlation, but we want the measurements to be close enough together that they represent a stationary underlying propensity. There is no single correct sampling procedure; it depends on theoretical considerations for the specific application domain.

As another example with real data, consider the time between successive failures of the air conditioning system in each aircraft in a fleet of Boeing 207 jet airplanes (Proschan, 1963). The time between successive failures is of interest so that managers can appropriately schedule preventative maintenance and maintain adequate inventory of spare parts. Thus, the emphasis in this situation is estimation of inter-failure duration. We will assume that successive failures of the same air conditioner are independent of each other. One rationale for this assumption is that the repair of a failed system resets the system so that it has no memory for the previous failure. The next failure depends only on other factors, such as the basic design of the air conditioner and the typical ambient conditions in which the individual aircraft flies.

The data from 12 aircraft are displayed in Figure 15.7 (these data omit one aircraft that had only two observations, as was done by Hand et al., 1994, #480). Notice that the data have been transformed from their original scale of days to days$^{1/5}$, i.e., the fifth root of days. This transformation was necessary because the raw data were extremely skewed and I wanted to use a normal likelihood function, for illustrative purposes (Proschan, 1963, used a more theoretically informed model). A different transformation, such as a logarithm, could have been used instead, but the fifth root yielded nicely symmetric data distributions. Notice that all the data were transformed the same way; there were not different transformations applied to different aircraft. As will be explained when the full model is described, the goal of the transformation was to make the data from each individual aircraft be approximately

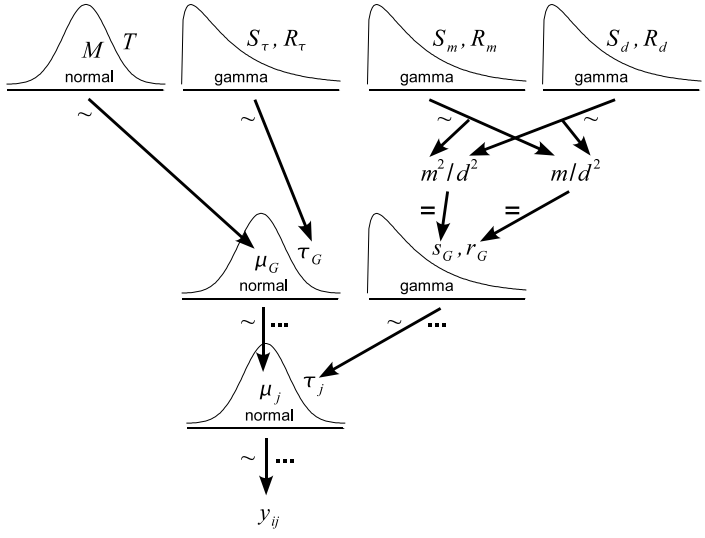

Figure 15.8: A model of hierarchical dependencies for repeated measures from each of several subjects drawn from the same group. The i^{th} datum from the j^{th} subject is denoted y_{ij}. It is generated by a normal distribution that has a subject mean of μ_j and subject precision of τ_j. The subject means are generated by a normal distribution centered on the overall mean of the group μ_G with a precision of τ_G. These group-descriptive parameters μ_G and τ_G, are themselves being estimated from the data, hence they have prior distributions as shown at the left of the hierarchy. The subject precisions τ_j, are generated by a gamma distribution which has parameters s_G and r_G. These parameters are themselves being estimated, therefore they have prior distributions as shown in the top right of the diagram.

normal, and to simultaneously make the distribution of all the means be approximately normal.

15.2.1 Hierarchical model

To model this situation, we will assume that data from the j^{th} individual are normally distributed around an underlying individual mean, denoted μ_j. The dispersion of repeated measurements from that individual is described by the precision τ_j. We will further assume that the individual propensities μ_j, are normally distributed around a group mean μ_G. The dispersion of the individual means around the group mean is described by the precision τ_G. It is these group-level parameters in which we are primarily interested. Because we want to estimate μ_G and τ_G, we need to specify their prior-belief distributions.

Just as the individual propensities μ_j are generated by group-level distributions, the individual precisions τ_j are also generated by group-level distributions. The idea is that the variability of repeated measurements in an individual is dependent on the group they are in. For example, a company might engender employees with very stable blood pressures, i.e., all the employees tend to have high precisions in their individual distributions of blood pressure. As another example, an air-conditioner manufacturer might build systems that fail very erratically, i.e., all the systems tend to have low precisions in their individual distributions of inter-failure duration. Therefore the individual τ_j come from a group-level gamma distribution which has shape and rate parameters we want to estimate from the

data. Therefore the shape and rate parameters of the group-level gamma distribution must themselves be given higher level prior distributions.

The entire model is illustrated in Figure 15.8. It shows the hierarchical dependency of individual parameters λ_j on the group-level parameters μ_G, r_G, and the dependency of the individual parameters λ_j on the group-level parameters μ_G, r_G. The group-level parameters themselves have higher-level distributions as shown. The higher level distributions are necessary because it is the group-level parameters we want to estimate, and therefore we must express our prior uncertainty in those group-level parameters by the higher-level distributions. The higher-level distributions for μ_G, r_G are re-parameterized in terms of the mean (m) and standard deviation (s) of the gamma density for λ_j. This reparameterization is performed merely for convenience in interpreting the values of μ_G, r_G.

At the top level, the constants for the hyper-priors must be given specific values. These values must be appropriate for the domain and data at hand. In other words, the hyper-prior constants should be at least mildly informed by the situation (or richly informed if audience-agreeable prior information is available). For example, we know from common sense that air conditioners will be manufactured to operate without failure for a duration on the order of 10^2 days, with a range of perhaps 10^0 to 10^4 days. Therefore the hyperprior on μ_G should reasonably specify a distribution in this range (and not seconds or eons!). Moreover, it is important to remember that the air-conditioner data were transformed to a different scale, and therefore the hyperprior constants need to be appropriate for the transformed scale.

Before proceeding with parameter estimation, it is important to verify that the basic distributional assumptions of the model are not being badly violated. At the lowest level of the model, the normal likelihood function assumes that the data generated by each individual aircraft are normally distributed. A quick look at Figure 15.7 shows that the twelve individual aircraft have data that are reasonably unimodal and symmetric (unlike the raw, untransformed data). At the next higher level in the model, the normal distribution that generates λ_j assumes that the central tendencies of the aircraft are normally distributed. Inspection of the top-middle graph of Figure 15.7 verifies that the 12 aircraft means are unimodal and symmetric. Finally, the gamma distribution at the level that generates the λ_j assumes that the precisions of the 12 aircraft can be described by a gamma distribution. Inspection of the top-right graph of Figure 15.7 verifies that the 12 aircraft precisions may be reasonably described this way.

If there appear to be prominent outliers in the data, then the data might be better modeled by a t distribution than by a normal distribution. As was discussed in connection with Figure 15.5, the t distribution is far less sensitive to outliers than the normal. The novel point here is that outliers might occur at different levels. There might be outliers of interfailure durations within each individual aircraft. These would be visible in the plots of individual aircraft data. Additionally, there might be outliers at the higher level of means for each aircraft. These outliers would be visible in the plot of aircraft means. In the hierarchical model of Figure 15.8, these considerations imply that either or both of the two lower normal distributions, involving λ_j or μ_G, might be replaced with a t distribution.

15.2.2 Implementation in BUGS

It is straightforward to implement the model in BUGS. Every row in the dependency diagram of Figure 15.8 has a corresponding specification in the BUGS model: (SystemsBrugs.R)

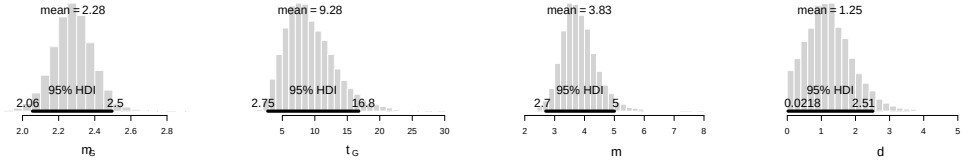

Figure 15.9: Posterior distribution for the top-level hyperparameters in Figure 15.8, for the data in Figure 15.7.

```
model {
    for( i in 1 : Ndata ) {
        y[i] ~ dnorm( mu[ subj[i] ] , tau[ subj[i] ] )
    }
    for ( j in 1 : Nsubj ) {
        mu[j] ~ dnorm( muG , tauG )
        tau[j] ~ dgamma( sG , rG )
    }
    muG ~ dnorm( 2.3 , 0.1 )
    tauG ~ dgamma( 1 , .5 )
    sG <- pow(m,2) / pow(d,2)
    rG <- m / pow(d,2)
    m ~ dgamma( 1 , .25 )
    d ~ dgamma( 1 , .5 )
}
```

Notice the use of nested indexing in line 11. In the BUGS model specification, the index i is the overall row of the data matrix, not the i^{th} measure within a subject. Nested indexing is used because the data matrix is structured with each row containing a datum y and the subject number (i.e., aircraft) from which the datum was obtained. This data structure accommodates the fact that different aircraft contributed different numbers of data points. The complete program is listed in Section 15.4.2.1 (SystemsBrugs.R).

Aspects of the resulting posterior distribution are shown in Figure 15.9. The left panel shows the marginal distribution on the group-level parameter μ_G, which has a posterior mean of 2.28 days$^{1=5}$, which corresponds to 61.6 days. In other words, on average, about two months elapsed between failures of the the air conditioning system on these airplanes. In the second panel, the value of $\sigma_G = 1/\sqrt{\tau_G}$ indicates the standard deviation across aircraft of the average inter-failure duration. The posterior mean of σ_G is 9.32, corresponding to a standard deviation of $1/\sqrt{9.32} = 0.33$ days$^{1=5}$. The parameters μ_G and σ_G describe the distribution of aircraft means μ_j, and thus the posterior is saying that among the most credible descriptions of the data distribution in the top-middle panel of Figure 15.7 is a normal distribution with mean 2.28 and standard deviation of 0.33. The remaining posterior marginals in Figure 15.9 show credible values of m and d, for the gamma distribution that describes the how the precisions vary across aircraft. The posterior mean of m is 3.83 and the posterior mean of d is 1.27, which says that among the most credible descriptions of the data distribution in the top-right panel of Figure 15.7 is a gamma distribution with mean of 3.83 and standard deviation of 1.27.

15.3 Summary

This chapter was intended to explore the normal likelihood function. The purpose was to explore concepts focused on the normal likelihood when it is simply a constant, not involving additional predictors and parameters. With this foundation in place, subsequent chapters can focus on concepts regarding additional predictors. This chapter made several main points:

- When the precision of the normal likelihood function is known (not estimated), then the "update rules" for the mean and precision of a normal prior are analytically simple. These formulas also reveal why it is convenient for BUGS to parameterize a normal distribution by its precision instead of by its standard deviation.
- It is typical to put a normal prior on the mean of a normal likelihood, and a gamma prior on the precision of a normal likelihood, because of considerations of conjugacy for simple situations. Use of these priors is not necessary, however. It is especially easy to program arbitrary priors is BUGS (although not all priors will work equally efficiently for sampling).
- When there are repeated measures within individuals, either the repeated measures within individuals and/or the means across individuals might be modeled as normal distributions. Again, this hierarchical model can be easily programmed in BUGS.
- Whenever applying a normal likelihood function, it is important to check that the data are reasonably normally distributed, i.e., at least roughly unimodal and symmetric. Otherwise the parameter values may be misleading. If the data are severely non-normal, they might be transformed to normal, or else a non-normal likelihood function could be used instead.
- Data that have prominent outliers might be better modeled with a t distribution than a normal distribution. This idea was introduced in this chapter, but examples of applying it in BUGS wait until subsequent chapters.

15.4 R code

15.4.1 Estimating the mean and precision of a normal likelihood

Notice that the data in this program are randomly generated from a normal distribution in R, on line 30. In R, dnorm and rnorm are parameterized by mean and standard deviation, unlike BUGS, in which dnorm is parameterized by mean and precision.

The program also initializes the chains intelligently so that they start amidst the posterior, instead of starting randomly according to the prior. This method of initializing chains was introduced in Section 9.5.2.2 (conBrugs.R). Lines 51–52 show that the initial value of is the mean of the data, and the initial value of is the precision of the data. If the prior is fairly vague, and the data are numerous, then the posterior will be near the parameter values that maximize the likelihood of the data. Therefore we start the chains near those maximum-likelihood values. We still need to allocate a burn-in period, however, because the chains needs to independently diverge from each other and settle in to the true posterior.

(YmetricXsingleBrugs.R)

```
graphics.off()
rm(list=ls(all=TRUE))
library(BRugs)              # Kruschke, J. K. (2010). Doing Bayesian dat      a analysis:
                              # A Tutorial with R and BUGS. Academic Press / Elsevier.
#------------------------------------------------- ----------------------------
# THE MODEL.
modelstring = "
# BUGS model specification begins here...
model {
    # Likelihood:
    for( i in 1 : N ) {
        y[i] ~ dnorm( mu , tau ) # tau is precision, not SD
    }
    # Prior:
    tau ~ dgamma( 0.01 , 0.01 )
    mu ~ dnorm( 0 , 1.0E-10 )
}
# ... end BUGS model specification
" # close quote for modelstring
writeLines(modelstring,con="model.txt")
modelCheck( "model.txt" )

#------------------------------------------------- ----------------------------
# THE DATA.

# Generate random data from known parameter values:
set.seed(47405)
trueM = 100
trueSD = 15
y = round( rnorm( n=500 , mean=trueM , sd=trueSD ) ) # R dnorm us es mean and SD

datalist = list(
    y = y ,
    N = length( y )
)

# Get the data into BRugs: (default filename is data.txt).
modelData( bugsData( datalist ) )

#------------------------------------------------- ----------------------------
# INTIALIZE THE CHAINS.

nchain = 3
modelCompile( numChains = nchain )

automaticInit = F # TRUE or FALSE
if ( automaticInit ) {
    modelGenInits()  # automatically initialize chains from pr          ior
} else {
    genInitList <- function() { # manually initialize chains ne           ar the data
        list( mu = mean( datalist$y ) ,
              tau = 1 / sd( datalist$y )^2 )
    }
    for ( chainIdx in 1 : nchain ) {
        modelInits( bugsInits( genInitList ) )
    }
}

#------------------------------------------------- ----------------------------
```

```
# RUN THE CHAINS

# burn in
BurnInSteps = 500
modelUpdate( BurnInSteps )
# actual samples
samplesSet( c( "mu" , "tau" ) )
stepsPerChain = 2000
thinStep = 1
modelUpdate( stepsPerChain , thin=thinStep )

#-------------------------------------------------- ----------------------------
# EXAMINE THE RESULTS

filenamert = "YmetricXsingleBrugs"

source("plotChains.R")
muSum = plotChains( "mu" , saveplots=F , filenamert )
sigmaSum = plotChains( "tau" , saveplots=F , filenamert )

muSample = samplesSample( "mu" )
tauSample = samplesSample( "tau" )
sigmaSample <- 1 / sqrt( tauSample ) # Convert precision to SD

source("plotPost.R")
windows()
plotPost( muSample , xlab="mu" , breaks=30 , main="Posteri or" )
dev.copy2eps(file=paste(filenamert,"PostMu.eps",sep ==""))

nPts = length(muSample) ; nPtsForDisplay = min( nPts , 2000 )
thinIdx = seq( 1 , nPts , nPts / nPtsForDisplay )
windows()
plot( muSample[thinIdx] , sigmaSample[thinIdx] , col="gr ay" ,
      xlab="mu" , ylab="sigma" , cex.lab=1.5 , main="Posterior" , log="y" )
points( mean(muSample) , mean(sigmaSample) , pch="+" , cex =2 )
text( mean(muSample) , mean(sigmaSample) ,
      bquote( .(round(mean(muSample),1)) *" "* .(round(mean(s igmaSample),1)) ),
      adj=c(.5,-0.5) )
dev.copy2eps(file=paste(filenamert,"PostMuSigma.eps ",sep=""))

#-------------------------------------------------- ----------------------------
```

15.4.2 Repeated measures: Normal across and normal within

Below is the complete program for analyzing the data from Fig 15.7, regarding inter-failure durations of aircraft air conditioners. A glimpse of the posterior was shown in Figure 15.9.

(SystemsBrugs.R)

```
graphics.off()
rm(list=ls(all=TRUE))
library(BRugs)          # Kruschke, J. K. (2010). Doing Bayesian dat a analysis:
                          # A Tutorial with R and BUGS. Academic Press / Elsevier.
#-------------------------------------------------- ----------------------------
# THE MODEL.
modelstring = "
# BUGS model specification begins here...
model {
```

```
      for( i in 1 : Ndata ) {
        y[i] ~ dnorm( mu[ subj[i] ] , tau[ subj[i] ] )
      }
      for ( j in 1 : Nsubj ) {
        mu[j] ~ dnorm( muG , tauG )
        tau[j] ~ dgamma( sG , rG )
      }
      muG ~ dnorm( 2.3 , 0.1 )
      tauG ~ dgamma( 1 , .5 )
      sG <- pow(m,2) / pow(d,2)
      rG <- m / pow(d,2)
      m ~ dgamma( 1 , .25 )
      d ~ dgamma( 1 , .5 )
}
# ... end BUGS model specification
" # close quote for modelstring
writeLines(modelstring,con="model.txt")
modelCheck( "model.txt" )

#------------------------------------------------- ----------------------------
# THE DATA.

# Load the aircraft data:
load( "Systems.Rdata" ) # loads dataMat
nSubj = length( unique( dataMat,"Aircraft"] ) )
# Transform the data:
DaysTransf = dataMat,"Days"]^(1/5)
dataMat = cbind( dataMat , DaysTransf )
colnames( dataMat ) = c( colnames( dataMat )[1:3] , "DaysTra      nsf" )

# Specify data, as a list.
datalist = list(
      y = dataMat,"DaysTransf"] ,
      subj = dataMat,"Aircraft"] ,
      Ndata = NROW(dataMat) ,
      Nsubj = nSubj
)
# Get the data into BRugs: (default filename is data.txt).
modelData( bugsData( datalist ) )

#------------------------------------------------- ----------------------------
# INTIALIZE THE CHAINS.

# First, compile the model:
nchain = 10
modelCompile( numChains = nchain )

modelGenInits() # works when the priors are not too flat

#------------------------------------------------- ----------------------------
# RUN THE CHAINS

# burn in
BurnInSteps = 1000
modelUpdate( BurnInSteps )
# actual samples
samplesSet( c( "muG" , "tauG" , "mu" , "tau" , "m" , "d" ) )
stepsPerChain = ceiling(10000/nchain)
thinStep = 100
```

```
69  modelUpdate( stepsPerChain , thin=thinStep )
70
71  #------------------------------------------------                    ----------------------------
72  # EXAMINE THE RESULTS
73
74  source("plotChains.R")
75  source("plotPost.R")
76  filenamert = "SystemsBrugs"
77
78  # Examine chains for convergence and autocorrelation:
79  muSum = plotChains( "muG" , saveplots=F , filenameroot=fil      enamert )
80  tauSum = plotChains( "tauG" , saveplots=F , filenameroot=f      ilenamert )
81  mSum = plotChains( "m" , saveplots=F , filenameroot=filena      mert )
82  dSum = plotChains( "d" , saveplots=F , filenameroot=filena      mert )
83  mu1Sum = plotChains( "mu[1]" , saveplots=F , filenameroot=      filenamert )
84  tau1Sum = plotChains( "tau[1]" , saveplots=F , filenameroo      t=filenamert )
85
86  # Extract chains from BUGS into R:
87  muGsample = samplesSample( "muG" )
88  tauGsample = samplesSample( "tauG" )
89  mSample = samplesSample( "m" )
90  dSample = samplesSample( "d" )
91  muSample = NULL
92  tauSample = NULL
93  for ( sIdx in 1:nSubj ) {
94        muSample = rbind( muSample , samplesSample( paste("mu[",s Idx,"]",sep="") ) )
95        tauSample = rbind( tauSample , samplesSample( paste("tau[     ",sIdx,"]",sep="") ) )
96  }
97
98  # Plot the aircraft mu:
99  windows(15,6)
100 layout( matrix( 1:nSubj , nrow=2 , byrow=T ) )
101 for ( sIdx in 1:nSubj ) {
102       plotPost( muSample[sIdx,] , xlab=bquote(mu[.(sIdx)]) )
103 }
104 dev.copy2eps(file=paste(filenamert,"PostMu.eps",sep         =""))
105
106 # Plot the aircraft tau:
107 windows(15,6)
108 layout( matrix( 1:nSubj , nrow=2 , byrow=T ) )
109 for ( sIdx in 1:nSubj ) {
110       plotPost( tauSample[sIdx,] , xlab=bquote(tau[.(sIdx)])              , HDItextPlace=.3 )
111 }
112 dev.copy2eps(file=paste(filenamert,"PostTau.eps",se         p=""))
113
114 # Plot the hyperdistributions:
115 windows(15,3)
116 layout( matrix(1:4,ncol=4) )
117 plotPost( muGsample , xlab=expression(mu[G]) , breaks=30      )
118 plotPost( tauGsample , xlab=expression(tau[G]) , breaks=       30 )
119 plotPost( mSample , xlab=expression(m) , breaks=30 )
120 plotPost( dSample , xlab=expression(d) , breaks=30 , HDIte      xtPlace=.1 )
121 dev.copy2eps(file=paste(filenamert,"PostHyper.eps",          sep=""))
122
123 #------------------------------------------------                    ----------------------------
```

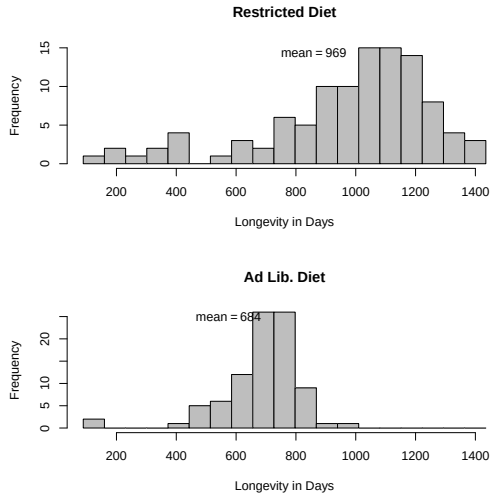

Figure 15.10: For Exercise 15.3.

15.5 Exercises

Exercise 15.1. [Purpose: View the prior from BUGS.] For the program in Section 15.4.1 (YmetricXsingleBrugs.R), generate graphs of prior distribution, analogous to Fig. 15.3. To do this, comment out the data in the data specification, as explained in Section 8.5.1, p. 141. But also beware of the following: Automatically initialize the chains from the prior, and, when plotting the results, comment out the plotChains commands because the MCMC sigma values are too extreme for some of the BUGS graphics routines.

Exercise 15.2. [Purpose: A realistic example of estimating a single mean, with consideration of priors.] University students who agreed to participate in a probability learning experiment were also tested for their vocabulary level, using the Wechsler Adult Intelligence Scale Revised (WAIS-R), which is normed to have a general-population mean of 10 and standard deviation of 3. Here are the data from the university students (data from Hand et al., 1994, set #392, p. 322):

14 11 13 13 13 15 11 16 10 13 14 11 13 12 10 14 10 14 16 14 14 11 11 11 13 12 13 11 11 15 14 16 12 17 9 16 11 19 14 12 12 10 11 12 13 13 14 11 11 15 12 16 15 11

(A) Are the data roughly normal, i.e., essentially unimodal and symmetrically distributed? (No formal analysis is required here; only an "eyeball" assessment is expected.) Therefore, can a normal likelihood function be applied?

(B) Discuss the rationale for a prior distribution. Justify the constants you choose for the hyperprior. Hint: The results of the analysis will be presented to a skeptical audience, who may doubt any claims about how prior knowledge of the general population informs an analysis of university students. Therefore your prior should be very vague and widely dispersed around the general population values.

(C) Is the mean vocabulary score of the university students credibly different from the general population mean of 10? Report the 95% HDI for the various priors you considered in the previous part.

Exercise 15.3. [Purpose: A realistic example of estimating a mean, with consideration of whether a normal likelihood distribution is appropriate.] Suppose we know that the mean life of a rat that eats ad lib is 700 days. This value is, in fact, about right for lab rats. See Figure 15.10, which shows data from R. L. Berger et al. (1988), as reported in Hand et al. (1994, data set #242), and which are available in the le RatLives.Rdata . This is an Rdata le, which stores values in compressed format, not text format. To get the data into R, type load("RatLives.Rdata") . It will look like nothing happens when you type that command, but the variables are now loaded into R; you can see the variables that R knows about by typing ls() . You will see listed the two vectors adlibDiet and restrictedDiet . When rats are placed on a restricted diet, their longevity can be affected. Figure 15.10 shows the results.

(A) Using the raw data from the restricted-diet longevities, estimate and its 95% HDI. Be sure to report the prior you used. Hint: The HDI extends from about 917 to 1020 days.

(B) Is it appropriate to apply a normal likelihood function to these data? Transform the data by squaring the longevities. Estimate and its 95% HDI (now in days squared). Be careful to use an appropriate prior! Hint: The HDI extends from about 967^2 to 1053^2 days squared.

(C) Why is the first estimate lower than the second estimate? Which estimate is more appropriate and why?

Exercise 15.4. [Purpose: Think about specifying an informed prior, and non-normal, non-gamma priors.] In the program of Section 15.4.1 (OneMetricXsingleBrugs.R), the prior on the group mean uses an extremely small precision. This diffuse prior is silly, because we know that individual IQ scores from the general population should be near 100, with a standard deviation of about 15. Special populations might have IQ scores consistently above or below 100, but almost certainly within, say, 100 points above or below 100. Consider the following expressions of the prior belief on :

(i) mu ~ dnorm(100 , sd=50) # that's sd=50, so tau=pow(50,-2)

(ii) mu ~ dgamma(4 , 0.04)

(iii) mu ~ dunif(0 , 200)

(iv) mu ~ dgamma(3 , 0.03)

(A) Plot the densities of (i) and (ii) on the same graph, superimposed. Hint:

```
mu = seq( -50 , 300 , length=501 )
plot( mu , dnorm( mu , 100 , 50 ) , type="l" , ylim=c(0,0.01) )
lines( mu , dgamma( mu , 4 , 0.04 ) )
```

What do the densities of (i) and (ii) have in common? (Hint: Compute the mean and sd of the gamma distribution.) Should either of (i) or (ii) be preferred over the other? (Hint: Consider negative IQ scores, which should not be allowed.)

(B) Plot the densities of (iii) and (iv) on the same graph, superimposed. What do the densities of (iii) and (iv) have in common? Should one of (iii) or (iv) be preferred over the other? (Hint: Consider IQ scores greater than 200, which should not be allowed.)

In the program of Section 15.4.1 (OneMetricXsingleBrugs.R), the prior on the group precision is a general diffuse distribution. This diffuse prior is silly, because we know that individual IQ scores have a standard deviation around 15, (a precision of $1/15^2 = 0.00444\ldots$) in the general population. The SD might be a bit smaller with special populations.

(C) Suppose we believe that the smallest SD we would find in a special population is 5. What is the corresponding precision? (Notice that it is larger than $1/15^2$.)

(D) We want to create a gamma distribution for precision that has a mean corresponding

to $1/15^2$, such that most of the gamma distribution is less than this value corresponding to an IQ precision of $1/15^2$, because, as mentioned in the previous part, we don't think that precisions much greater than $1/15^2$ are tenable. Therefore, to be sure that our prior encompasses that maximum but still allows for considerable uncertainty, we will set the standard deviation of the gamma distribution to half of the difference between the precisions determined in the previous two parts. What is the value of this standard deviation? (Hint: The answer is 0.01777... Explain.)

(E) What are the values of the shape and rate parameters for a gamma distribution that has mean of 0.00444 and standard deviation of 0.01777?

(F) Re-run the program of Section 15.4.1 (`OneMetricXsingleBrugs.R`) using a prior for tau from the previous part, and a prior for mu that is the most appropriate from (i)–(iv) above. Include the posterior histogram of mu with your answer. Does the conclusion differ noticeably from the diffuse priors?

Exercise 15.5. [Purpose: With repeated measures, the group estimate and individual estimates reflect different sources of variation in the data.]

Consider the data, y, with different assignments to subjects, s, as follows:

y values:	1	2	3	21	22	23	41	42	43
s, large between subj. var.:	1	1	1	2	2	2	3	3	3
s, small between subj. var.:	1	2	3	1	2	3	1	2	3

The first row shows nine data values. The second row indicates a situation in which the first three data points come from subject 1, the second three data points come from subject 2, and so on. The third row indicates a different situation, in which the first, fourth, and seventh data points come from subject 1, as so forth. The assignment of data to subjects in the second row produces large between-subject variance, whereas the assignment of data to subjects in the third row produces small between-subject variance. Notice that for both situations, however, the overall mean is the same because the data are the same. We are interested in which situation gives a more precise estimate of the group-level mean.

Modify the program of Section 15.4.2 (`SystemsBrugs.R`) for use with these data. (No need to be too fancy. For example, in the `dataList`, just type in `y = c( 1, 2, 3, 21, 22, 23, 41, 42, 43 )` and `subj = c( 1,1,1, 2,2,2, 3,3,3 )`.) Be sure to make the prior appropriate for these data. Run the program twice, once for each assignment of data to subjects (using the same prior for both runs). Hint: Your results should look something like those in Figure 15.11. Notice that the mean of the posterior of α_0 is essentially the same for both runs, but the HDI widths are quite different. Explain why.

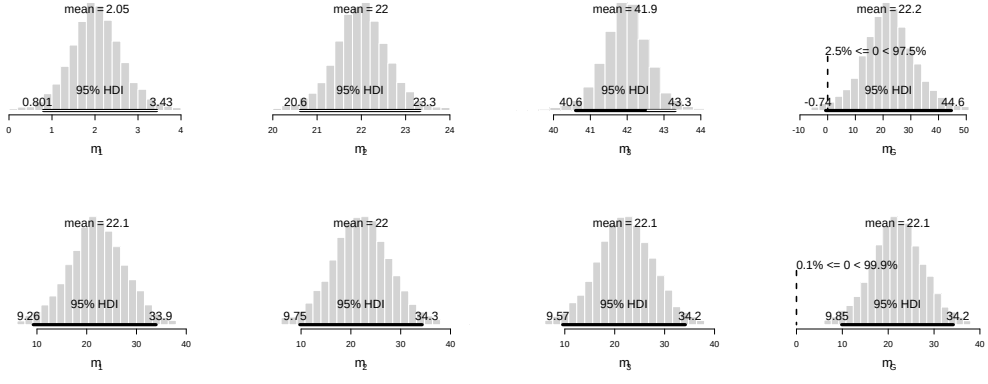

Figure 15.11: For Exercise 15.5, involving the same data arranged across subjects. Top row: Large between-subject variance. Bottom row: Small between-subject variance. Notice the width of the HDI on the group level parameter, μ_G.

Chapter 16

Metric Predicted Variable with One Metric Predictor

Contents

> 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?

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. We will describe the relationship between the predicted variable, y, and predictor, x, with a simple linear model and normally distributed residual randomness in y. In terms of the generalized linear model (GLM), the model assumes that $y \sim N(\mu, \sigma)$ with $\mu = \beta_0 + \beta_1 x$. This model appears in Table 14.1 (p. 312) in the first row and second column. This model is often referred to as "simple linear regression".

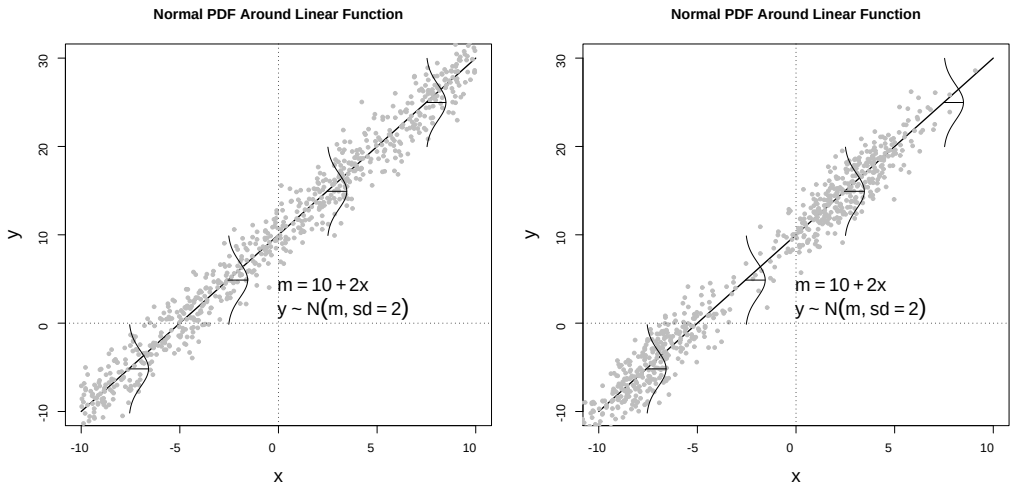

Figure 16.1: Examples of points normally distributed around a linear function. (The left panel repeats Figure 14.9, p. 309.) The model assumes that the data are normally distributed vertically around the line, as shown. Moreover, the variance of y is the same at all values of x. The model puts no constraints on the distribution of x. The right panel shows a case in which x is distributed bimodally, whereas in the left panel the x are distributed uniformly. In both panels, there is homogeneity of variance.

16.1 Simple linear regression

Figure 16.1 shows an example of simulated data generated from the assumed model, with parameters displayed in the Figure. At any value of x, the mean predicted value is $\mu = \beta_0 + \beta_1 x$, and the data values y are normally distributed around that mean. For a review of how to interpret the slope, β_1, and intercept, β_0, see Figure 14.1, p. 295.

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 16.1 were sampled randomly from a uniform distribution, merely for purposes of illustration, whereas the x values in the right panel of Figure 16.1 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 16.1 because the x values are uniformly distributed: The smattering of data points in the vertical, y, direction appears visually to be the same at all values of x. Homogeneity of variance is less easy to identify visually with x values of not uniformly distributed. For example, the right panel of Figure 16.1 displays data that may appear to violate homogeneity of variance, because the apparent vertical spread of the data seems to be larger for $x = 2.5$ than for $x = 7.5$. 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 far less opportunity for the sampled y values to come from the tails of the normal 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

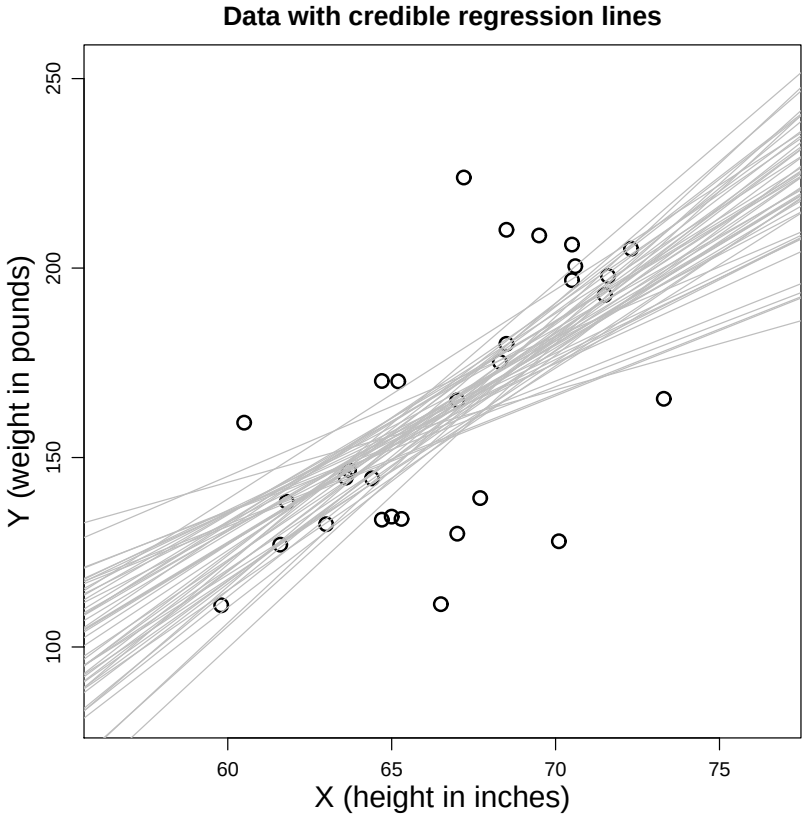

Figure 16.2: Data points, with a smattering of believable regression lines superimposed.

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 any of all of those! 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.

As an example, suppose we have measurements of height, in inches, and weight, in pounds, for some randomly selected people. Figure 16.2 shows the height and weight values of the people. The data were generated from the process listed in Section 16.5.1 (HtWtDataGenerator.R), which uses realistic population parameters. The data points in Figure 16.2 do appear to indicate that as height increases, weight also tends to increase. This covariation between height and weight does not imply that one attribute causes the other. When an adult eats a lot of sugary foods, or goes on a diet, he or she changes his or her weight, his or her height does not change. Despite the lack of direct causal relationship, the two values do covary, and one can be (imperfectly) predicted from the other.

Our goal is to determine what regression lines are most believable, given the data. In other words, we want to infer what combinations of $_0$, $_1$, and $$ are most believable, given the data. We use Bayes' rule:

$$p(_0, _1, |y) = \frac{p(y|_0, _1,)\, p(_0, _1,)}{\int d_0\, d_1\, d\; p(y|_0, _1,)\, p(_0, _1,)}$$

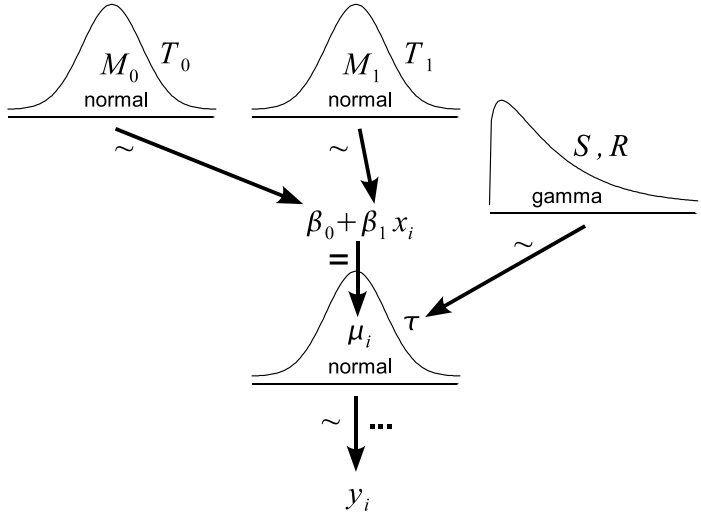

Figure 16.3: A model of dependencies for simple linear regression. The parameter β_1 is of primary interest; it describes the slope of the line relating x to y.

Analytical forms for the posterior can be obtained for appropriate priors. For example, analogous to the derivation in the previous chapter, if the precision is fixed, i.e., if the prior on τ is a spike over a certain value, then normal priors on the slope and intercept yield normal posteriors (e.g., Bolstad, 2007). Fortunately, we do not have to worry much about mathematical derivations because we can let BUGS approximate the posterior. Our job, therefore, is to specify sensible priors and to make sure that BUGS generates a trustworthy posterior sample that is converged and well-mixed.

16.1.1 The hierarchical model and BUGS code

There are three parameters in the model, and therefore we need to establish a prior distribution on the joint three-dimensional parameter space. To keep things simple, we will assume that the three parameters are independent in our priors. They would not need to be; for example, we might have a prior belief that the slope and intercept are (anti-)correlated, or we might have a prior belief that the precision and the intercept are correlated. For now, however, we assume independence in the prior, and therefore we can express the prior as three marginal distributions on the separate parameters.

Figure 16.3 shows the hierarchy of dependencies that we will assume. At the bottom of the figure, we see that the data depend on the parameters μ and τ, which describe the mean and precision of a normal distribution. In other words, the data are modeled by a normal likelihood function, as was emphasized in the previous chapter. The value of the precision, τ, depends on our prior belief distribution, shown in the upper right of the figure as a gamma distribution with two shape parameters. The shape parameters are constants that express what we think the precision is likely to be and how uncertain we are in that prior belief.

The mean, μ, of the likelihood function is a linear function of x, as shown in the middle of Figure 16.3. The linear function has two parameters, the intercept β_0 and the slope β_1. The slope parameter is what we are usually most interested in, because it describes the relation between x and y. The values of the slope and intercept depend on the prior belief shown in the upper layer of the figure. Both the slope and the intercept here have prior

beliefs modeled as normal distributions.

It is useful to compare Figure 16.3 with Figure 15.2, p. 322. The underlying approach is the same for both scenarios, in that there is a normal likelihood for the data, and we set priors on the mean and the precision. The only evidence is that Figure 16.3 has one additional prior for the $_1$ parameter. Otherwise, the models are identical.

Every arrow in Figure 16.3 has a corresponding line in the BUGS model speci cation: (SimpleLinearRegressionBrugs.R)

```
model {
    for( i in 1 : Ndata ) {
        y[i] ~ dnorm( mu[i] , tau )
        mu[i] <- beta0 + beta1 * x[i]
    }
    beta0 ~ dnorm( 0 , 1.0E-12 )
    beta1 ~ dnorm( 0 , 1.0E-12 )
    tau ~ dgamma( 0.001 , 0.001 )
}
```

Notice that the constants in the hyperpriors are generic values for a di use prior. In real applications, you would probably want to use better informed constants.

16.1.1.1 Standardizing the data for MCMC sampling

In principle, we could run the BRugs code on the raw x, y data. In practice, however, the attempt often fails. There's nothing wrong with the mathematics or logic, the problem is that believable values of the slope and intercept parameters tend to be tightly correlated, and this narrow diagonal zone of believability is di cult for sampling algorithms to explore.

The right panel of Figure 16.4 shows an example of this type of correlation in believable values. The believable slopes and intercepts are extremely correlated. Sampling from such a tightly correlated distribution is typically very di cult to do directly. It is di cult to discover a point in the narrow zone in the rst place. Then, having discovered a viable point, the chain does not move e ciently. Gibbs sampling gets stuck because it keeps "bumping into the walls" (recall discussion of Section 8.4.2.1, p. 213). Metropolis algorithms often are not clever enough to automatically tune a proposal distribution to match a diagonal posterior.

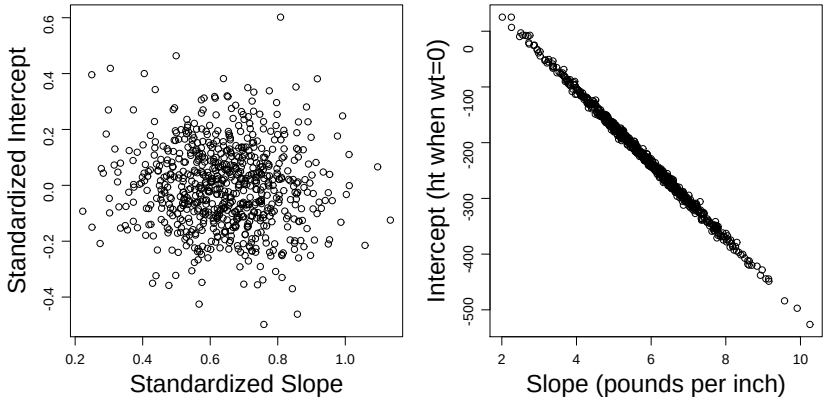

Figure 16.4: Slope and intercept values of believable regression lines, in standardized and raw scales.

The reason for the correlation of slope and intercept is evident by examining Figure 16.2. There you can see various believable lines that go through the scatter of points. Notice that if a line has a steep slope, its intercept must be low, but if a line has a smaller slope, its intercept must be higher. Thus, there is a trade-off in slope and intercept for the believable lines.

One of the main tricks used for successful execution of the MCMC sampling is standardizing the data. Standardizing simply means re-scaling the data relative to their mean and standard deviation:

$$z(x) = \frac{(x - M_x)}{SD_x} \quad \text{and} \quad z(y) = \frac{y - M_y}{SD_y} \tag{16.1}$$

where M_x is the mean of the data x values and SD_x is the standard deviation of the data x values. (Do not confuse M_x and M_y with the constants used in the specification of the priors in the hierarchical diagram.) 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.

Having used BUGS to find slope and intercept values for the standardized data, we then need to convert the parameter values back to the original scales. Denote the intercept and slope for standardized data as ζ_0 and ζ_1 (Greek letter "zeta"), and denote the predicted value of y as $\hat{y}$. Then:

$$\begin{aligned} z_{\hat{y}} &= \zeta_0 + \zeta_1 z_x && \text{by definition of the model} \\ \frac{\hat{y} - M_y}{SD_y} &= \zeta_0 + \zeta_1 \frac{(x - M_x)}{SD_x} && \text{from Eqn. 16.1} \\ \hat{y} &= \left\{\zeta_0 SD_y + M_y - \zeta_1 SD_y M_x / SD_x\right\} + \left\{\zeta_1 SD_y / SD_x\right\} x \end{aligned} \tag{16.2}$$

Thus, for every believable combination of ζ_0, ζ_1 values, there is a corresponding believable combination of β_0, β_1 values specified by Equation 16.2.

The sampled points in Figure 16.4 were generated by running JAGS on the standardized data, then transforming the results according to Equation 16.2. Notice that the standardized slope and intercept, in the left panel of Figure 16.4, show no noticeable correlation. This is because the believable intercepts tend to be near zero, and that's because the standardized data have means of zero. Thus, even when the slopes of believable lines differ, the intercepts still hover around zero.

16.1.1.2 Initializing the chains

Figure 16.4 showed an example of how the believable values in the posterior distribution can occupy a fairly narrow region of parameter space. For the MCMC chain to randomly sample from the posterior, the random walk must first get into the modal region of the posterior in the first place. We might simply start the chain at any point in parameter space, randomly selected from the prior distribution, and wait through the burn-in period until the chain randomly wanders into the bulk of the posterior. Unfortunately, for many real-world situations, this burn-in period can be very long time. Therefore it helps to initialize the chains near the bulk of the posterior if we can.

If the data set is large and dominates the prior, or if the prior is diffuse, or if the prior is informed from previous results and is reasonably consistent with the new data, then the peak of the likelihood function will be reasonably near the peak of the posterior distribution. Therefore, a reasonable candidate for the initial point of the chain is the maximum-likelihood estimate of the parameters. This heuristic is useful if we have a simple way of determining the maximum-likelihood estimate. Fortunately, that is the case for simple linear regression, as we do. When the data are standardized, the maximum-likelihood estimate (MLE) of $_0$ is zero, the MLE of $_1$ is the correlation (denoted r) of x and y, and the MLE of the precision, , is $1 \quad 1 \quad r^2$. To get an intuition for the statement about precision, consider what happens when the correlation r approaches 1 (its maximum possible value). As r approaches 1, the $x;y$ data fall very close to a straight line, which implies that the deviation of the data away from the line is very small (recall Figure 16.1). Hence when r is large, is small, and hence is large, as reflected by the formula $1 \quad 1 \quad r^2$.

16.1.2 The posterior: How big is the slope?

Figure 16.5 shows the posterior distribution of slope values. The standardized and original-scale slopes indicate the same relationship on different scales, and therefore the posterior distributions are identical except for a change of scale. The posterior distribution tells us exactly what we want to know: The believable slopes. We see that weight increases by about 5 or 6 pounds for every 1-inch increase in height. The 95% HDI provides a useful summary of the range of believable slopes.

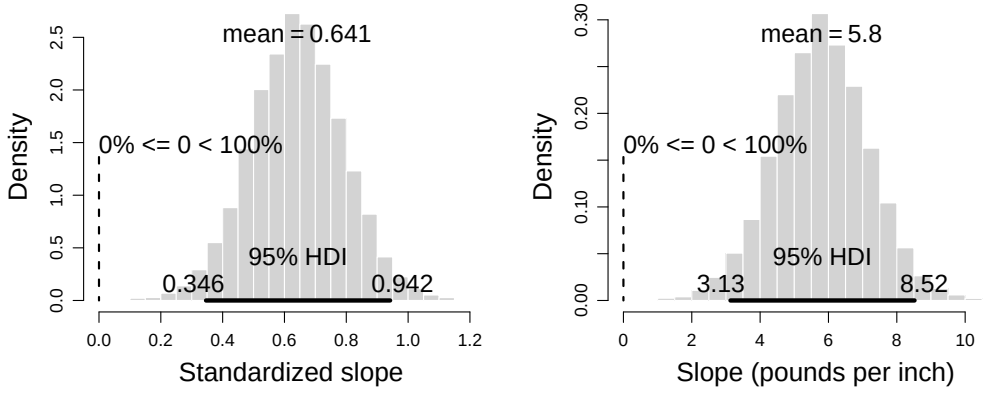

Figure 16.5: Posterior distribution of slopes.

If we were interested is determining whether the predictor had a non-zero influence on the predicted variable, we might use the decision rules discussed in Section 12.1.3 (p. 244). We may want to establish a ROPE around zero for the predictor, and then check whether the entire 95% HDI excludes the ROPE. (Usually, if the true value is zero, the HDI will overlap the ROPE, thereby reducing false alarms.)

It would be possible to "test" whether the slope is non-zero by doing a Bayesian comparison of two models: One model would have an arbitrarily wide prior on the slope parameter; the other model would have an arbitrarily peaked prior at zero. It would be straightforward, in principle, to set up the model comparison using trans-dimensional MCMC as described in Section according to the method explained in Sec. 10.2 (p. 197). But, as argued in Section 12.2.2 (p. 249) and elsewhere, all that model comparison tells us

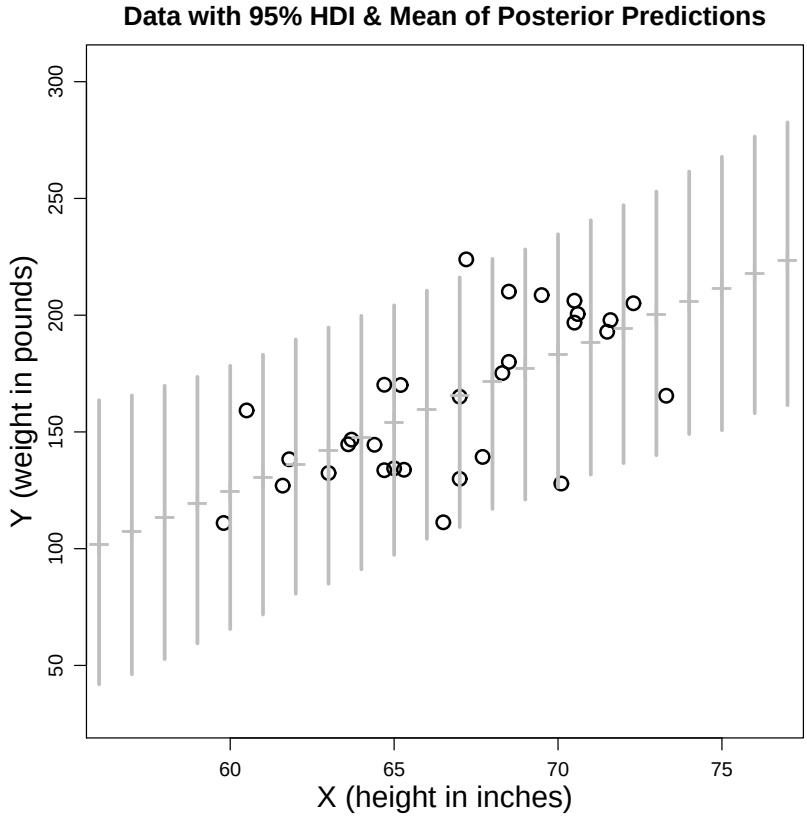

Figure 16.6: Data points, with 95% HDIs and means of posterior predictions in y, at selected x values. The HDIs are wider (i.e., taller) for extrapolated x values than for x values in the middle of the observed data.

is which of two unbelievable models is less unbelievable. In the present application, the model comparison would favor the model that allows non-zero slopes. But that is not all we want to know, because usually we also want to know what slope values are credible. That information is provided by the posterior distribution on the parameter values, often starting with an (mildly) informed prior distribution instead of an arbitrarily diffuse prior of mathematical convenience. Thus, in the present application, there is little to be gained by doing a Bayesian model comparison of null prior against arbitrary diffuse prior.

16.1.3 Posterior prediction

Linear regression is often used for predicting y values from x values. For example, we can use the results of the regression to answer the question, If a person were 75 inches tall, what is the probable weight of the person? The Bayesian answer provides the probability of every possible weight y, given the height x and the previous data: $p(y|x, D)$. This distribution of y values has uncertainty stemming from the inherent noise and also from uncertainty in the estimated values of the regression coefficients.

A simple way to get a good approximation of $p(y|x, D)$ is by generating random values of y for every step in the MCMC sample of credible parameter values. Thus, at any step in the chain, there are particular values of β_0, β_1, and σ, which we use to generate representative predicted values of y according to $y \sim N(\mu = \beta_0 + \beta_1 x, \sigma = 1/\sqrt{\tau})$. We do that at all

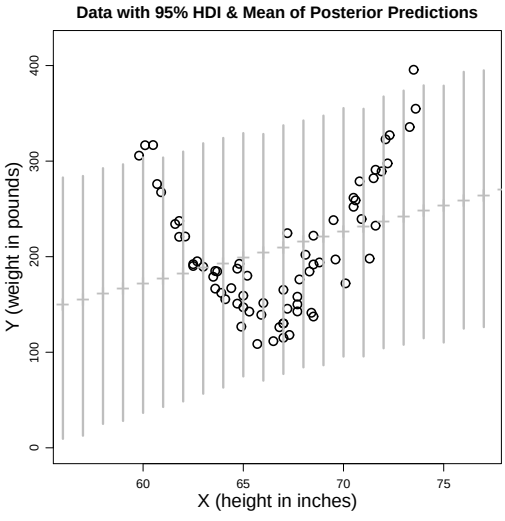

Figure 16.7: Data points, with 95% HDIs and means of posterior predictions, at selected x values. The data show systematic discrepancy from the linear-model predictions.

the steps in the chain, and thereby create a large set of credible y values. From this set, we can compute the mean predicted y value, the HDI of the predicted y values, etc.

A summary of the posterior predictions is displayed in Figure 16.6. Each vertical grey segment, over a particular x value, indicates the extent of the 95% HDI for the distribution of randomly generated y values at that x value. The dash across the middle of the grey segment indicates the mean of the posterior predicted y values.

Consider the limits of the HDIs at each value of x. Notice that the length of the HDIs is larger for high and low x values than for middling x values. To verify this fact, hold a ruler against the HDIs for small, middling, and large values of x. This variation in the length of the HDIs makes sense intuitively: As the x value goes farther from the observed data, the predicted y value should become less certain. Another way of understanding the variation in HDI lengths is by considering the credible regression lines in Figure 16.2. All of those different sloped lines go near the bulk of the data in the middle, but the different slopes extend higher or lower at extrapolative values of x. We average across all those lines to generate predicted y values. Therefore the predicted y values will be more widespread at extrapolative values of x.

If the model of the data is a good model of the data, then credible parameter values for the model ought to generate simulated data that "look like" the real data. What it means for simulated data to "look like" the real data is defined by the analyst, and may be anything that is useful for the application at hand. Loosely speaking, a model is a good model if (1) the data fall mostly within the predicted zone, (2) the data are distributed approximately the way the model says they should be, e.g., normally, and (3) the discrepancies of data from predictions are random and not "systematic".

Posterior prediction is also useful for alerting us to nonlinear trends in data. Figure 16.7 shows a set of data for which the y values have a clear quadratic (i.e., parabolic) relationship with x, in addition to a linear trend. These data can be entered into the linear regression program, and the resulting posterior prediction HDIs are shown by the vertical grey segments in Figure 16.7. We see that the data do fall mostly within the range of the posterior predictions, but the distribution of the data is systematically discrepant from the linear spine of the model: At high and low values of x, the data fall well above the linear spine, but at middling values of x, the data fall well below the linear spine. This sort of systematic discrepancy from the posterior predictions is a clue that the model could be improved.

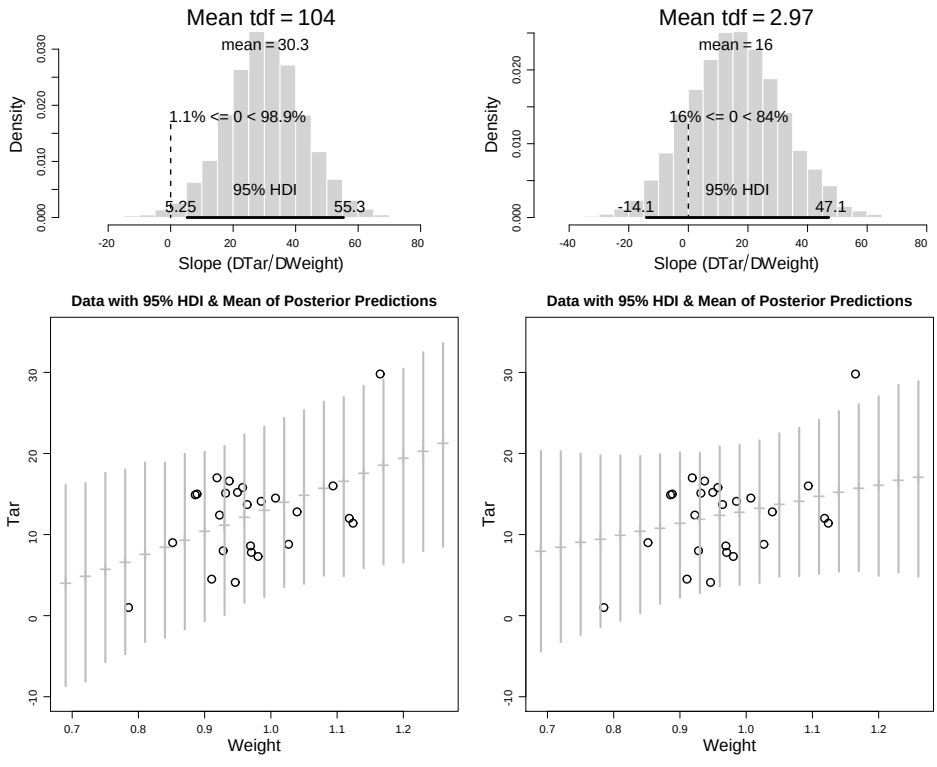

Figure 16.8: Data fit with different priors on the ν parameter of the t likelihood function. Left column shows results from prior bias on high ν values, such that a virtually normal likelihood function is assumed. Right column shows "robust" regression resulting from a strong prior bias on low ν values.

16.2 Outliers and robust regression

Recall from Figure 15.5, p. 325, that the estimated parameter values for a normal likelihood function can be greatly distorted by outliers in the data. This sensitivity also occurs when the normal distribution is used in linear regression. The normal likelihood function demands that the regression line is vertically close to all the data points, because the likelihood value is tiny for points more than about three standard deviations from the line. Consequently, outlying data points can have disproportionate leverage in determining the estimate of the regression coefficients.

As an example, consider some data regarding 25 brands of cigarette (McIntyre, 1994). For each brand, a cigarette was assessed for its weight, amount of nicotine, amount of tar, and amount of carbon monoxide produced when burned. Many different relationships among these variables might be investigated, but consider in particular predicting the tar content from the weight of the cigarette. These data are plotted in the lower panels of Figure 16.8. The points are rather diversely scattered, and do not show an overwhelmingly strong covariation. There is, however, an outlying point that has the heaviest weight and highest tar content (in the upper right of the scatterplot). This outlying point can have a disproportionate influence on the estimated slope (β_1) in the regression model.

The left column of Figure 16.8 shows the results when a normal likelihood function is used (actually, when the likelihood function is a t distribution with large ν). The histogram in the upper panel shows that the slope is credibly larger than zero. The plot of posterior

predictive HDIs in the lower panel shows that the estimated regression line has been tilted up toward the outlier. The attraction to the outlier is caused by the small tails of the normal likelihood function: For the outlying datum to be "under the umbrella" of the normal likelihood, the line must get fairly close.

The right column of Figure 16.8 shows the results when a t distribution is used as the likelihood function, with a prior on the ν parameter that strongly favors small values. This prior implies that the likelihood function has the tall tails of a low-df t distribution, unless the data strongly suggest otherwise. As can be seen in the figure, the 95% HDI of the posterior on the slope contains zero. The lower right panel shows that the estimated regression line is not slanted as far toward the outlier, because the tall tail of the t distribution gives the outlier some modest non-zero probability despite being so far from the mean of the distribution. The posterior slope estimate here better reflects the bulk of the points in the scatterplot, and is not so strongly dominated by the outlier.

The BUGS model for robust regression is a simple extension of the model for the normal likelihood. We simply replace the normal distribution with a t distribution, and include the necessary prior specification. Whereas the normal distribution has two parameters, namely the mean and the precision, the t distribution has those two plus a third parameter, namely the degrees of freedom, df, which can take on real values of 1 or greater (for a reminder of how the df parameter works, see Figure 15.4, p. 324). One way to put a prior on the df parameter is by sampling a value, denoted udf, from a uniform distribution, and then transforming that value into the range allowed for df, which extends from 1 to infinity. One such transformation appears on line 18 of this model specification: (SimpleRobustLinearRegressionBrugs.R)

```
9  model {
10     for( i in 1 : Ndata ) {
11         y[i] ~ dt( mu[i] , tau , tdf )
12         mu[i] <- beta0 + beta1 * x[i]
13     }
14     beta0 ~ dnorm( 0 , 1.0E-12 )
15     beta1 ~ dnorm( 0 , 1.0E-12 )
16     tau ~ dgamma( 0.001 , 0.001 )
17     udf ~ dunif(0,1)
18     tdf <- 1 - tdfGain * log(1-udf) # tdf in [1,Inf).
19     # tdfGain specified in data section
20 }
```

Notice that tdf is used as the df parameter in the t likelihood function specified on line 11. The value of tdf is determined from udf by the transformation tdf <- 1 - tdfGain * log(1 - udf), where tdfGain is a constant that expresses the prior belief in large values of df. Figure 16.9 shows a graph of the transformation. You can see from the graph that when tdfGain is small, the tdf values are close to 1 across almost the entire range of udf. But when tdfGain is large, then the tdf values are large across most of the range of udf.

The prior on the df parameter expresses how much we believe there are outliers in the data. For example, the left column of Figure 16.8 was created by setting tdfGain=100, and the right column of Figure 16.8 was created by setting tdfGain=1. The resulting posteriors are noticeably different.

So, you may ask, which conclusion from Figure 16.8 is correct? Does the tar content of cigarettes increase by about 30 units for every unit increase in weight, as indicated by the nearly-normal likelihood, or does the tar content increase by about only about half that much, as indicated by the low-df t distribution? The answer is, in this case, that it depends

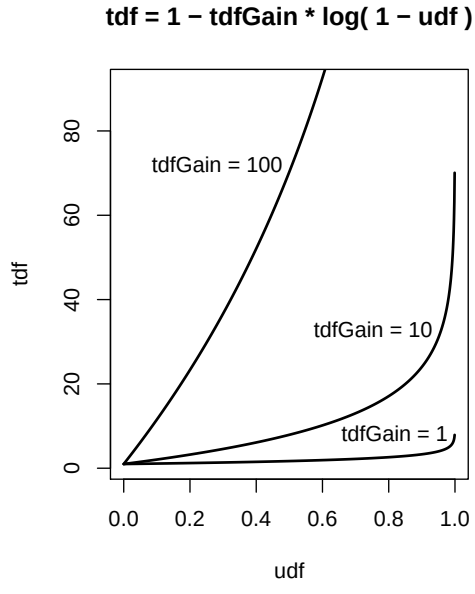

Figure 16.9: Transformation from u_{df} to t_{df}, used to set the prior of the t_{df} parameter in robust regression.

on your prior beliefs. If you have some very good reason to believe that tar content should be normally distributed (with the same variance) at every level of weight, then the left column of Figure 16.8 describes your posterior beliefs. But the assumption of normality virtually disavows the possibility of outliers in the data, and therefore seems untenable, at least for this application. Therefore the prior should allow low values of ν. The dependency of the posterior on the prior suggests that either more specific prior knowledge should be brought to bear, or additional predictors should be included if available, or more data need to be collected.

16.3 Simple linear regression with repeated measures

Suppose that for every individual j, we have multiple observations of x_{ij}, y_{ij} pairs. With these data, we can estimate a linear model for every individual. If we also assume that the individuals are mutually representative of a common group, then we can use the estimates from the individuals to inform estimates of group-level parameters.

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 model can describe each child's reading ability through time, and higher-level distributions describe the distribution of intercepts and slopes across individuals. By virtue of being linked directly through the higher level distribution, estimates of individuals are mutually informed by other individual.

As another concrete example of this situation, consider an experiment in which the investigators were interested in how quickly different organs clear themselves of contaminants (Feldman, 1988, who reported data from an unpublished experiment by S. B. Weinstock and J. D. Brain). The researchers administered iron oxide particles to rats, because the iron oxide remaining in the body could be assayed noninvasively via magnetometry. Four rats were given intravenous injection of iron oxide, particles of which are taken up by liver endothelial cells. Four other rats were given the iron oxide by tracheal instillation, so that the particles were taken up by lung macrophages. Although the researchers were

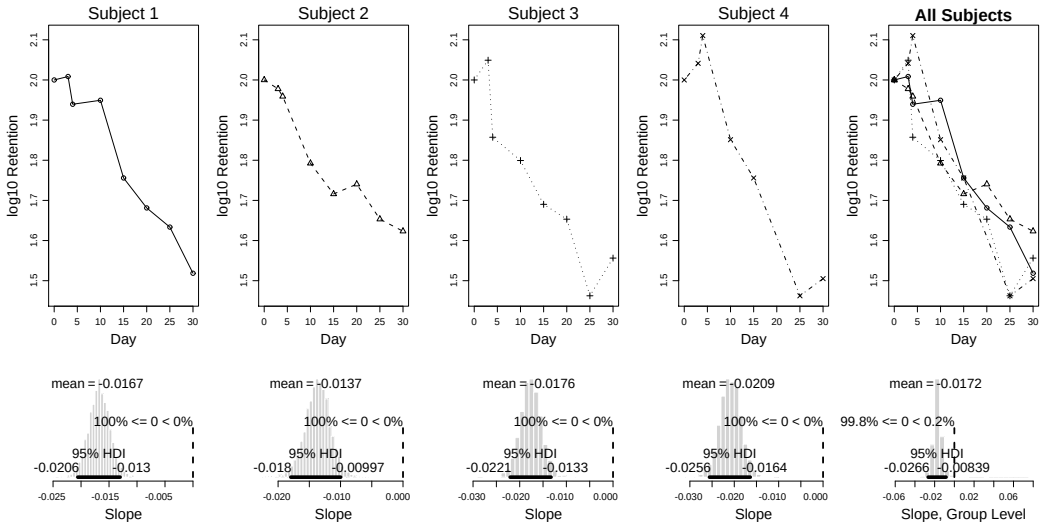

Figure 16.10: Upper row shows data from Feldman (1988). Lower row shows believable values of the slopes.

speci cally interested in comparing the clearance rates of the two groups, we will consider only the “lung” group.

The amount of iron oxide retained in the body, as a percentage of the initially assayed amount, was measured at various times during the following 30 days. The data for the lung group are shown in the top row of Figure 16.10. The retention amount is plotted on a logarithmic scale, so that the retention curves are approximately linear. Notice that all the curves start at a y value of 2.0, because the first measurement establishes the baseline that defines 100%, and $\log_{10}(100) = 2.0$. Some curves rise above the initial value, which presumably does not indicate spontaneous creation of iron, but instead indicates measurement noise, either at the initial or subsequent times. The graphs indicate a reduction through time. The goal of our analysis is to determine what the believable rates of reduction are, given the data. We may also want to know if the apparent reduction is believably non-zero.

To model this situation, each subject's data set is described by an individual linear regression, and the regression coefficients of the individuals are, in turn, modeled by group-level distributions. The group-level distributions are controlled by parameters that describe the central tendency and variability of the group, and these are the group-level parameters in which we are primarily interested.

Figure 16.11 shows the hierarchy of dependencies. At the bottom level, we just see a simple linear regression for each individual, with the same components as Figure 16.3. The regression coefficients for the j^{th} individual, namely β_{0j}, β_{1j}, and ν_j, in turn come from distributions that describe the group. For example, the individual slope coefficients, β_{1j}, are assumed to come from a normal distribution, with mean μ_{1G} and precision τ_{1G}. We are interested in estimating those group-level parameters, and therefore each is given a prior distribution at the top level of the diagram.

Every arrow in Figure 16.11 has a corresponding line in the BUGS model specification: (SimpleLinearRegressionRepeatedBrugs.R)

```
8 model {
9     for( r in 1 : Ndata ) {
```

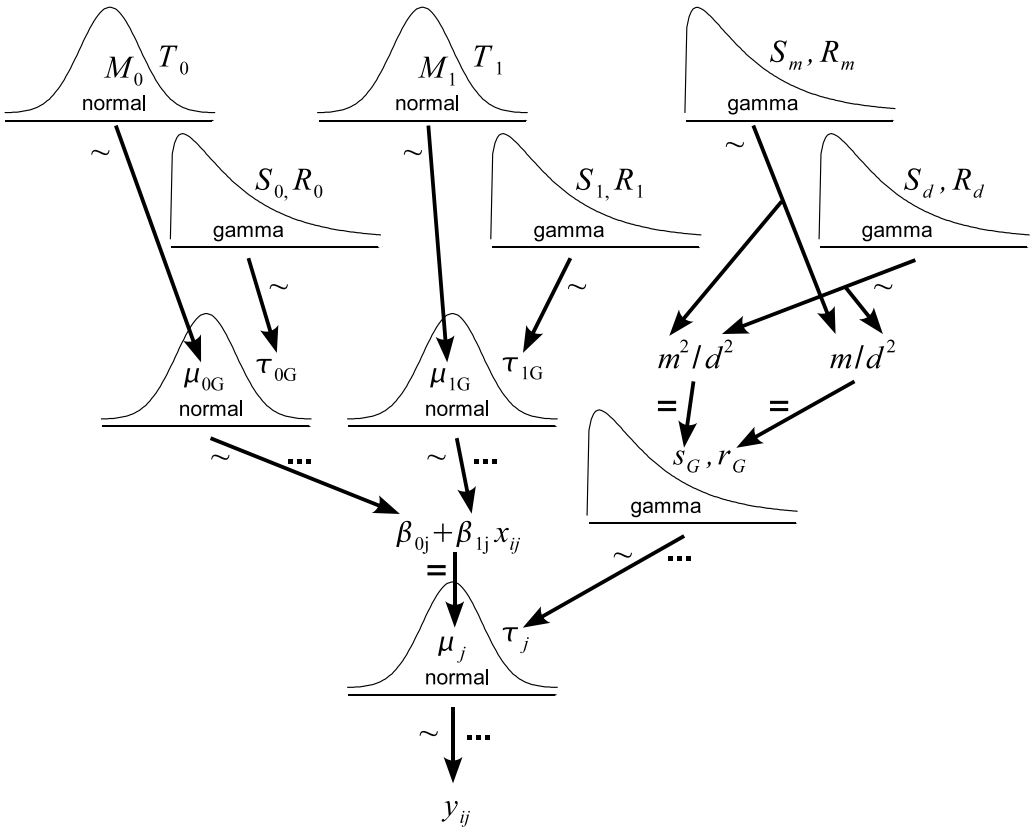

Figure 16.11: A model of dependencies for repeated scores on N subjects drawn independently from the same group. The slope for the j^{th} subject is β_{1j}. Across subjects, it is distributed normally, with mean μ_{1G} and precision τ_{1G}, i.e., across subjects the variance of the slopes is $1/\tau_{1G}$.

```
        y[r] ~ dnorm( mu[r] , tau[ subj[r] ] )
        mu[r] <- b0[ subj[r] ] + b1[ subj[r] ] * x[r]
    }
    for ( s in 1 : Nsubj ) {
        b0[s] ~ dnorm( mu0G , tau0G )
        b1[s] ~ dnorm( mu1G , tau1G )
        tau[s] ~ dgamma( sG , rG )
    }
    mu0G ~ dnorm(0,.01)
    tau0G ~ dgamma(.1,.1)
    mu1G ~ dnorm(0,.01)
    tau1G ~ dgamma(.1,.1)
    sG <- pow(m,2)/pow(d,2)
    rG <- m/pow(d,2)
    m ~ dgamma(1,.1)
    d ~ dgamma(1,.1)
}
```

The constants in the top-level priors are generically vague because of laziness during programming. More thoughtful priors may be desirable.

The results of running the program are shown in the lower row of Figure 16.10. The results match closely those reported in the non-Bayesian analysis of Feldman (1988), who reported 0:0173 for the group slope. The results also show that every individual slope is

believably different from zero, as is the estimate of the group average slope.

The four rats in this experiment had remarkably similar clearance rates. In many realms of research, there is much more variation between subjects. Suppose that different rats had very different clearance rates, with some faster and some slower than the rats reported here, such that the group-average clearance rate was about the same. What aspects of the posterior would be most affected by the increase in variability of individual slopes? Would the certainty of individual slope estimates be much affected? Would the certainty of the group-average slope be much affected? Exercise 16.1 has you systematically alter the data, to answer these questions. It is also a good intuition-building exercise to consider the influence of differing individual intercepts. In particular, without changing the individual slopes, but merely by changing the individual intercepts, the estimate of the group-average slope can be affected. See Exercise 16.2.

The data in this example were very well behaved, insofar as there were no blatantly outlying data points. Specifically, within each individual, all the values descended in a reasonably linear trend. And, across subjects, all the individual slopes were reasonably similar to each other, with no subjects who had clearances that were much faster or much slower than the others. With a larger sample, however, outlying subjects might be encountered. And there is always the chance that device malfunctions or transcription errors could be inadvertently introduced into individual data points. Therefore whenever there is reason to believe that outliers may be lurking in the data, t distributions can be substituted for the normal distributions at the appropriate level. Exercise 16.3 shows an example of outlying slopes and intercepts, involving income and family size.

16.4 Summary

This chapter explained Bayesian simple linear regression, where "simple" means it involves a single predictor. There were several main points:

- Prediction of y from x does not imply or assume any particular causal or temporal relation between y and x.
- The usual model for linear regression assumes homogeneity of variance: The variance of y is the same at all values of x.
- It is typical, but not necessary, to put normal priors on the intercept and slope parameters in the linear regression model.
- To make MCMC sampling efficient, it is helpful to standardize the y and x data, and to initialize the chains at the MLE values.
- Posterior predictions are useful for predicting y values from x values, especially when extrapolating or interpolating to novel x values. Posterior predictions are also useful for checking whether there are systematic discrepancies of the data from the predictions of the model.
- It is straightforward to extend the model to situations with repeated measures, in which every individual has data estimated by simple linear regression, and overarching parameters describe the distribution of slope, intercept, and precision parameters of the group.

When there are outliers in repeated measures from an individual, or in the distribution of parameters across individuals, a t distribution may be used instead of a normal distribution.

In the next chapter, we introduce additional predictors to the model, and explore the concept of interaction across predictors.

16.5 R code

16.5.1 Data generator for height and weight

(HtWtDataGenerator.R)

```
HtWtDataGenerator = function( nSubj , rndsd=NULL ) {
# Random height, weight generator for males and females. Uses parameters from
# Brainard, J. & Burmaster, D. E. (1992). Bivariate distributions for height and
# weight of men and women in the United States. Risk Analysis, 12(2), 267-275.
# John K. Kruschke, January 2008.

require(MASS)

# Specify parameters of multivariate normal (MVN) distributions.
# Men:
HtMmu = 69.18
HtMsd = 2.87
lnWtMmu = 5.14
lnWtMsd = 0.17
Mrho = 0.42
Mmean = c( HtMmu , lnWtMmu )
Msigma = matrix( c( HtMsd^2 , Mrho * HtMsd * lnWtMsd ,
                    Mrho * HtMsd * lnWtMsd , lnWtMsd^2 ) , nrow=2 )
# Women cluster 1:
HtFmu1 = 63.11
HtFsd1 = 2.76
lnWtFmu1 = 5.06
lnWtFsd1 = 0.24
Frho1 = 0.41
prop1 = 0.46
Fmean1 = c( HtFmu1 , lnWtFmu1 )
Fsigma1 = matrix( c( HtFsd1^2 , Frho1 * HtFsd1 * lnWtFsd1 ,
                    Frho1 * HtFsd1 * lnWtFsd1 , lnWtFsd1^2 ) , nrow=2 )
# Women cluster 2:
HtFmu2 = 64.36
HtFsd2 = 2.49
lnWtFmu2 = 4.86
lnWtFsd2 = 0.14
Frho2 = 0.44
prop2 = 1 - prop1
Fmean2 = c( HtFmu2 , lnWtFmu2 )
Fsigma2 = matrix( c( HtFsd2^2 , Frho2 * HtFsd2 * lnWtFsd2 ,
                    Frho2 * HtFsd2 * lnWtFsd2 , lnWtFsd2^2 ) , nrow=2 )

# Randomly generate data values from those MVN distributions.
if ( !is.null( rndsd ) ) { set.seed( rndsd ) }
datamatrix = matrix( 0 , nrow=nSubj , ncol=3 )
colnames(datamatrix) = c( "male" , "height" , "weight" )
maleval = 1 ; femaleval = 0 # arbitrary coding values
```

```
for ( i in 1:nSubj ) {
          # Flip coin to decide sex
          sex = sample( c(maleval,femaleval) , size=1 , replace=TRUE       , prob=c(.5,.5) )
          if ( sex == maleval ) {
         datum = mvrnorm(n = 1, mu=Mmean, Sigma=Msigma ) }
          if ( sex == femaleval ) {
         Fclust = sample( c(1,2) , size=1 , replace=TRUE , prob=c(pro      p1,prop2) )
         if ( Fclust == 1 ) {
              datum = mvrnorm(n = 1, mu=Fmean1, Sigma=Fsigma1 ) }
         if ( Fclust == 2 ) {
              datum = mvrnorm(n = 1, mu=Fmean2, Sigma=Fsigma2 ) }
    }
          datamatrix[ i , ] = c( sex , round( c( datum[1] , exp( datum[2] )          ) , 1 ) )
}

return( datamatrix )
} # end function
```

16.5.2 BRugs: Robust linear regression

(SimpleRobustLinearRegressionBrugs.R)

```
graphics.off()
rm(list=ls(all=TRUE))
library(BRugs)          # Kruschke, J. K. (2010). Doing Bayesian dat      a analysis:
                          # A Tutorial with R and BUGS. Academic Press / Elsevier.
#------------------------------------------------------                  ----------------------------
# THE MODEL.
modelstring = "
# BUGS model specification begins here...
model {
    for( i in 1 : Ndata ) {
        y[i] ~ dt( mu[i] , tau , tdf )
        mu[i] <- beta0 + beta1 * x[i]
    }
    beta0 ~ dnorm( 0 , 1.0E-12 )
    beta1 ~ dnorm( 0 , 1.0E-12 )
    tau ~ dgamma( 0.001 , 0.001 )
    udf ~ dunif(0,1)
    tdf <- 1 - tdfGain * log(1-udf) # tdf in [1,Inf).
    # tdfGain specified in data section
}
# ... end BUGS model specification
" # close quote for modelstring
writeLines(modelstring,con="model.txt")
modelCheck( "model.txt" )

#------------------------------------------------------                  ----------------------------
# THE DATA.

cigData = read.csv(file="McIntyre1994data.csv")
nSubj = NROW(cigData)
x = cigData[,"Wt"]
xName="Weight"
y = cigData[,"Tar"]
yName="Tar"

# Re-center data at mean, to reduce autocorrelation in MCMC s    ampling.
# Standardize (divide by SD) to make initialization easier.
```

```
xM = mean( x ) ; xSD = sd( x )
yM = mean( y ) ; ySD = sd( y )
zx = ( x - xM ) / xSD
zy = ( y - yM ) / ySD

# Specify data, as a list.
tdfGain = 1 # 1 for low-baised tdf, 100 for high-biased tdf
datalist = list(
  x = zx ,
  y = zy ,
  Ndata = nSubj ,
  tdfGain = tdfGain
)
# Get the data into BRugs:
modelData( bugsData( datalist ) )

#------------------------------------------------- ----------------------------
# INTIALIZE THE CHAINS.

nchain = 3
modelCompile( numChains = nchain )

genInitList <- function() {
    r = cor(x,y)
    list(
        beta0 = 0 ,     # because data are standardized
        beta1 = r ,          # because data are standardized
        tau = 1 / (1-r^2) , # because data are standardized
        udf = 0.95 # tdf = 4
    )
}
for ( chainIdx in 1 : nchain ) {
    modelInits( bugsInits( genInitList ) )
}

#------------------------------------------------- ----------------------------
# RUN THE CHAINS

# burn in
BurnInSteps = 100
modelUpdate( BurnInSteps )
# actual samples
samplesSet( c( "beta0" , "beta1" , "tau" , "tdf" ) )
stepsPerChain = ceiling(10000/nchain)
thinStep = 10
modelUpdate( stepsPerChain , thin=thinStep )

#------------------------------------------------- ----------------------------
# EXAMINE THE RESULTS

source("plotChains.R")

fname = paste("SimpleRobustLinearRegressionBrugsTdfGa  in",tdfGain,
                  sep="")
#beta0Sum = plotChains( "beta0" , saveplots=F , filenamero     ot=fname )
#beta1Sum = plotChains( "beta1"       , saveplots=F , filenamero     ot=fname )
#tauSum    = plotChains( "tau"          , saveplots=F , filenameroot=f       name )
#tdfSum    = plotChains( "tdf"          , saveplots=F , filenameroot=f       name )

```

```
# Extract chain values:
tdfSamp = samplesSample( "tdf" )
tdfM = mean( tdfSamp )
z0 = samplesSample( "beta0" )
z1 = samplesSample( "beta1" )
zTau = samplesSample( "tau" )
zSigma = 1 / sqrt( zTau ) # Convert precision to SD

# Convert to original scale:
b1 = z1 * ySD / xSD
b0 = ( z0 * ySD + yM - z1 * ySD * xM / xSD )
sigma = zSigma * ySD

# Posterior prediction:
# Specify x values for which predicted y's are needed:
xRang = max(x)-min(x)
yRang = max(y)-min(y)
limMult = 0.25
xLim= c( min(x)-limMult*xRang , max(x)+limMult*xRang )
yLim= c( min(y)-limMult*yRang , max(y)+limMult*yRang )
yLim = c(-10,35)
xPostPred = seq( xLim[1] , xLim[2] , length=20 )
# Define matrix for recording posterior predicted y values a t each x value.
# One row per x value, with each row holding random predicted y values.
postSampSize = length(b1)
yPostPred = matrix( 0 , nrow=length(xPostPred) , ncol=post SampSize )
# Define matrix for recording HDI limits of posterior predic ted y values:
yHDIlim = matrix( 0 , nrow=length(xPostPred) , ncol=2 )
# Generate posterior predicted y values.
# This gets only one y value, at each x, for each step in the chai n.
for ( chainIdx in 1:postSampSize ) {
  yPostPred[,chainIdx] = rnorm( length(xPostPred) ,
                                mean = b0[chainIdx] + b1[chainIdx] * xPostPred ,
                                sd = rep( sigma[chainIdx] , length(xPostPred) ) )
}
source("HDIofMCMC.R")
for ( xIdx in 1:length(xPostPred) ) {
  yHDIlim[xIdx,] = HDIofMCMC( yPostPred[xIdx,] )
}

# Display believable beta0 and b1 values
windows()
par( mar=c(4,4,1,1)+0.1 , mgp=c(2.5,0.8,0) )
#layout( matrix(1:2,nrow=1) )
thinIdx = seq(1,length(b0),length=700)
#plot( z1[thinIdx] , z0[thinIdx] , cex.lab=1.75 ,
#     ylab="Standardized Intercept" , xlab="Standardized Slo pe" )
plot( b1[thinIdx] , b0[thinIdx] , cex.lab=1.75 ,
      ylab="Intercept" , xlab="Slope" )
dev.copy2eps(file=paste(fname,"SlopeIntercept.eps", sep=""))

# Display the posterior of the b1:
source("plotPost.R")
windows(7,4)
par( mar=c(4,4,1,1)+0.1 , mgp=c(2.5,0.8,0) )
#layout( matrix(1:2,nrow=1) )
#histInfo = plotPost( z1 , xlab="Standardized slope" , comp Val=0.0 ,
#                     breaks=30 )
histInfo = plotPost( b1 , main=bquote("Mean tdf"==.(signi f(tdfM,3))) , cex.main=2 ,
```

```
                    xlab=bquote("Slope (" * Delta * .(yName) / Delta * .(xName)
                                          * ")") , compVal=0.0 , breaks=30 )
dev.copy2eps(file=paste(fname,"PostSlope.eps",sep="" ))

# Display data with believable regression lines and posteri or predictions.
windows()
par( mar=c(3,3,2,1)+0.5 , mgp=c(2.1,0.8,0) )
# Plot data values:
plot( x , y , cex=1.5 , lwd=2 , col="black" , xlim=xLim , ylim=y Lim ,
      xlab=xName , ylab=yName , cex.lab=1.5 ,
      main="Data with credible regression lines" , cex.main=1.3 3 )
# Superimpose a smattering of believable regression lines:
for ( i in seq(from=1,to=length(b0),length=50) ) {
    abline( b0[i] , b1[i] , col="grey" )
}
dev.copy2eps(file=paste(fname,"DataLines.eps",sep="" ))

# Display data with HDIs of posterior predictions.
windows()
par( mar=c(3,3,2,1)+0.5 , mgp=c(2.1,0.8,0) )
# Plot data values:
#yLim= c( min(c(yHDIlim,y)) , max(c(yHDIlim,y)) )
plot( x , y , cex=1.5 , lwd=2 , col="black" , xlim=xLim , ylim=y Lim ,
      xlab=xName , ylab=yName , cex.lab=1.5 ,
      main="Data with 95% HDI & Mean of Posterior Predictions" , ce x.main=1.33 )
# Superimpose posterior predicted 95% HDIs:
segments( xPostPred, yHDIlim[,1] , xPostPred, yHDIlim[,2 ] , lwd=3, col="grey" )
points( xPostPred , rowMeans( yPostPred ) , pch="+" , cex=2 , col="grey" )
dev.copy2eps(file=paste(fname,"DataPred.eps",sep="" ))

#------------------------------------------------- ----------------------------
```

16.5.3 BRugs: Simple linear regression with repeated measures

(SimpleLinearRegressionRepeatedBrugs.R)

```
graphics.off()
rm(list=ls(all=TRUE))
library(BRugs)              # Kruschke, J. K. (2010). Doing Bayesian dat a analysis:
                              # A Tutorial with R and BUGS. Academic Press / Elsevier.
#------------------------------------------------- ----------------------------
# THE MODEL.
modelstring = "
model {
    for( r in 1 : Ndata ) {
        y[r] ~ dnorm( mu[r] , tau[ subj[r] ] )
        mu[r] <- b0[ subj[r] ] + b1[ subj[r] ] * x[r]
    }
    for ( s in 1 : Nsubj ) {
        b0[s] ~ dnorm( mu0G , tau0G )
        b1[s] ~ dnorm( mu1G , tau1G )
        tau[s] ~ dgamma( sG , rG )
    }
    mu0G ~ dnorm(0,.01)
    tau0G ~ dgamma(.1,.1)
    mu1G ~ dnorm(0,.01)
    tau1G ~ dgamma(.1,.1)
    sG <- pow(m,2)/pow(d,2)
    rG <- m/pow(d,2)
```

```
24        m ~ dgamma(1,.1)
25        d ~ dgamma(1,.1)
26    }
27    " # close quote for modelstring
28    writeLines(modelstring,con="model.txt")
29    modelCheck( "model.txt" )
30
31    #------------------------------------------------- ----------------------------
32    # THE DATA.
33
34    # Data from H. A. Feldman, 1988, Table 4, p. 1731.
35    # Columns are "group" , "subjID" , "time" , "retention"
36    source("Feldman1988Table4data.R")
37    # Remove missing data:
38    includeRowVec = is.finite( Feldman1988Table4data[,"ret ention"] )
39    dataMat = Feldman1988Table4data[ includeRowVec , ]
40    # Retain only the Group 1 (lung) data:
41    dataMat = dataMat[ dataMat[,"group"]==1 , ]
42    # Convert to log10(retention):
43    dataMat[,"retention"] = log10( dataMat[,"retention"] )
44    # Column names and plot labels
45    yColName = "retention" ; yPlotLab = "log10 Retention"
46    xColName = "time" ; xPlotLab = "Day"
47    subjColName = "subjID" ; subjPlotLab = "Subject"
48    fname = "SimpleLinearRegressionRepeatedBrugs"
49
50    if ( F ) { # change to T to use income data instead of contam.rete ntion data.
51     # Data from http://www.census.gov/hhes/www/income/stat emedfaminc.html
52     # Downloaded Dec. 06, 2009.
53     load("IncomeFamszState.Rdata") # loads IncomeFamszStat e
54     dataMat = IncomeFamszState
55     yColName="Income" ; yPlotLab = "Income"
56     xColName="Famsz" ; xPlotLab="Family Size"
57     subjColName="State" ; subjPlotLab="State"
58     fname = "IncomeFamszState"
59    }
60
61    # Extract data info to pass to BUGS:
62    Ndata = NROW(dataMat)
63    subj = as.integer(factor(dataMat[,subjColName]))
64    Nsubj = length(unique(subj))
65    x = as.numeric(dataMat[,xColName])
66    y = as.numeric(dataMat[,yColName])
67
68    # Re-center data at mean, to reduce autocorrelation in MCMC s ampling.
69    # Standardize (divide by SD) to make initialization easier.
70    xM = mean( x ) ; xSD = sd( x )
71    yM = mean( y ) ; ySD = sd( y )
72    zx = ( x - xM ) / xSD
73    zy = ( y - yM ) / ySD
74
75    # Specify data, as a list.
76    datalist = list(
77      Ndata = Ndata ,
78      Nsubj = Nsubj ,
79      subj = subj ,
80      x = zx ,
81      y = zy
82    )
```

```
# Get the data into BRugs:
modelData( bugsData( datalist ) )

#------------------------------------------------                    ----------------------------
# INTIALIZE THE CHAINS.

nchain = 3
modelCompile( numChains = nchain )

genInitList <- function() {
    b0 = b1 = tau = rep(0,length=Nsubj)
    for ( sIdx in 1:Nsubj ) {
        yVec = datalist$y[datalist$subj==sIdx]
        xVec = datalist$x[datalist$subj==sIdx]
        lmInfo = lm( yVec ~ xVec )
        b0[sIdx] = lmInfo$coef[1]
        b1[sIdx] = lmInfo$coef[2]
        tau[sIdx] = length(yVec) / sum(lmInfo$res^2)
    }
    mu0G = mean(b0)
    tau0G = 1/sd(b0)^2
    mu1G = mean(b1)
    tau1G = 1/sd(b1)^2
    m = mean(tau)
    d = sd(tau)
    list( b0=b0 , b1=b1 , tau=tau ,
          mu0G=mu0G , tau0G=tau0G ,
          mu1G=mu1G , tau1G=tau1G ,
          m=m , d=d )
}
for ( chainIdx in 1 : nchain ) {
    modelInits( bugsInits( genInitList ) )
}

#------------------------------------------------                    ----------------------------
# RUN THE CHAINS

# burn in
BurnInSteps = 500
modelUpdate( BurnInSteps )
# actual samples
samplesSet( c( "b0","b1","tau" , "mu0G","tau0G", "mu1G",       "tau1G", "m","d" ) )
stepsPerChain = ceiling(5000/nchain)
thinStep = 100 # 40 or more
modelUpdate( stepsPerChain , thin=thinStep )

#------------------------------------------------                    ----------------------------
# EXAMINE THE RESULTS

source("plotChains.R")
source("plotPost.R")

# Check convergence and autocorrelation:
checkConvergence = T  # check this first time through, examin    e m,d,tau0G,tau1G
if ( checkConvergence ) {
    # check a few selected chains
    b01Sum    = plotChains( "b0[1]"   , saveplots=F , filenameroot=      fname )
    b11Sum    = plotChains( "b1[1]"   , saveplots=F , filenameroot=      fname )
    tau1Sum   = plotChains( "tau[1]" , saveplots=F , filenameroo       t=fname )
```

```
    mu0GSum  = plotChains( "mu0G"    , saveplots=F , filenameroot=  fname )
    tau0GSum = plotChains( "tau0G"   , saveplots=F , filenameroo    t=fname )
    mu1GSum  = plotChains( "mu1G"    , saveplots=F , filenameroot=  fname )
    tau1GSum = plotChains( "tau1G"   , saveplots=F , filenameroo    t=fname )
    mSum     = plotChains( "m"       , saveplots=F , filenameroot=fname )
    dSum     = plotChains( "d"       , saveplots=F , filenameroot=fname )
}

# Extract chain values for subsequent examination:
zmu0Gsamp = samplesSample( "mu0G" )
zmu1Gsamp = samplesSample( "mu1G" )
zb0samp = NULL
zb1samp = NULL
for ( subjIdx in 1:Nsubj ) {
    zb0samp = rbind( zb0samp , samplesSample( paste("b0[",sub  jIdx,"]",sep="") ))
    zb1samp = rbind( zb1samp , samplesSample( paste("b1[",sub  jIdx,"]",sep="") ))
}

# Convert to original scale:
mu0Gsamp = zmu0Gsamp * ySD + yM - zmu1Gsamp * ySD * xM / xSD
mu1Gsamp = zmu1Gsamp * ySD / xSD
b1samp   = zb1samp * ySD / xSD

# Display believable intercept and slope values
windows(10,5.5)
par( mar=c(4,4,1.75,1)+0.1 , mgp=c(2.5,0.8,0) )
layout( matrix(1:2,nrow=1) )
thinIdx = round(seq(1,length(mu0Gsamp),length=700))
plot( zmu1Gsamp[thinIdx] , zmu0Gsamp[thinIdx] , cex.lab=     1.75 ,
      ylab="Standardized Intercept" , xlab="Standardized Slop      e" )
plot( mu1Gsamp[thinIdx] , mu0Gsamp[thinIdx] , cex.lab=1.     0 ,
      ylab=paste("Intercept (",yPlotLab," when ",xPlotLab," =        0)",sep="") ,
      xlab=paste("Slope (change in",yPlotLab,"per unit",xPlo       tLab,")") )
dev.copy2eps(file=paste(fname,"SlopeIntercept.eps",       sep=""))

# Make graphs of data and corresponding believable slopes:
windows(12,6)
par( mar=c(4,4,1.75,1)+0.1 , mgp=c(2.5,0.8,0) )
layout(matrix(c(1:5,1:5,6:10),nrow=3,byrow=T))
xlims = c( min( dataMat[,xColName] ) ,  max( dataMat[,xColNa    me] ) )
ylims = c( min( dataMat[,yColName] ) ,  max( dataMat[,yColNa    me] ) )
sIdVec = unique( dataMat[,subjColName] )
# Plot data of individual subjects:
nSubjPlots = 4 # number of representative subject plots to ma     ke
subjIdxVec = round(seq(1,length(sIdVec),length=nSubjP      lots))
for ( sIdx in subjIdxVec ) {
    rVec = ( dataMat[,subjColName] == sIdVec[sIdx] )
    plot( dataMat[rVec,xColName] , dataMat[rVec,yColName] ,      type="o" ,
          ylim=ylims , ylab=yPlotLab , xlab=xPlotLab , cex.lab=1.5 ,
          pch=sIdx%%26 , lty=sIdx , main=bquote(.(subjPlotLab) *" "      * .(sIdx)) ,
          cex.main=1.75 )
}
# Plot data of all subjects superimposed
plot( NULL,NULL, xlab=xPlotLab,xlim=xlims , ylab=yPlotL      ab,ylim=ylims ,
      cex.lab=1.5 , main=paste("All ",subjPlotLab,"s",sep=""      ) , cex.main=1.75 )
for ( sIdx in 1:length(sIdVec) ) {
    rVec = ( dataMat[,subjColName] == sIdVec[sIdx] )
    lines( dataMat[rVec,xColName] , dataMat[rVec,yColName]      ,
           lty=sIdx , pch=sIdx%%26 , type="o")
```

```
}
# Plot histograms of corresponding posterior slopes:
for ( sIdx in subjIdxVec ) {
    histInfo = plotPost( b1samp[sIdx,] , xlab="Slope" , compVa    l=0.0 , breaks=30 ,
                         HDItextPlace=0.9 )
}
histInfo = plotPost( mu1Gsamp , xlab="Slope, Group Level" ,    compVal=0.0 ,
                         breaks=30 , HDItextPlace=0.9 )
dev.copy2eps(file=paste(fname,"Data.eps",sep=""))

#-------------------------------------------------    ----------------------------
```

16.6 Exercises

Exercise 16.1. [Purpose: See the influence of individual slope differences on the estimate of the group-average slope.]

The data shown in Figure 16.10 indicate that all subjects had similar rates of decline in retention, and therefore the estimate of the group average slope is fairly certain. In this exercise we change the data so that the individual slopes differ more dramatically, and examine the effect on the estimate of the group average.

(A) Alter the data as follows: In the program listed in Section 16.5.1 (SimpleLinearRegressionRepeatedBrugs.R), just before the data are renamed from dataMat to x and y (at about line 65), subtract 0.030 x from subject 1, subtract 0.015 x from subject 2, do nothing to subject 3, and add 0.015 x to subject 4. Here is an example of code for subject 1:

```
subjRowVec = ( dataMat[,subjColName] == 1 )
dataMat[ subjRowVec , yColName ] = ( dataMat[ subjRowVec , yC olName ]
                                     - .030 * dataMat[ subjRowVec , xColName ] )
```

Just repeat and modify for the remaining subjects. Run the program and include the plot of the data. (See Figure 16.12.) The data curves for the four subjects should have four very different slopes.

(B) Relative to the original data, has the posterior mean of the group slope gotten farther away from zero or closer to zero? Include the histogram of the posterior in your write-up. Are all the individual slopes believably different from zero (according to the 95% HDI)? Is the group slope believably different from zero (according to the 95% HDI)? Why is the group-level slope, which is now farther away from zero on average, less believably different from zero than in the original data?

Exercise 16.2. [Purpose: See the influence of differences in individual intercepts on the estimate of group-average slope.]

The data shown in Figure 16.10 all start at 2.0 because that is $\log_{10}(100)$, and the data were measured as percentage of original value. Suppose that the data were kept in their raw magnitudes, instead of converted to percentage of original magnitude. This would merely change the intercepts of the individual data curves, without changing their slopes, because $\log(ky) = \log(k) + \log(y)$. In this exercise we find out whether this change would have any effect on the estimate of the group slope.

(A) Alter the data as follows: In the program listed in Section 16.5.1 (SimpleLinearRegressionRepeatedBrugs.R), right before the data are converted to log10 on line 42, insert this code:

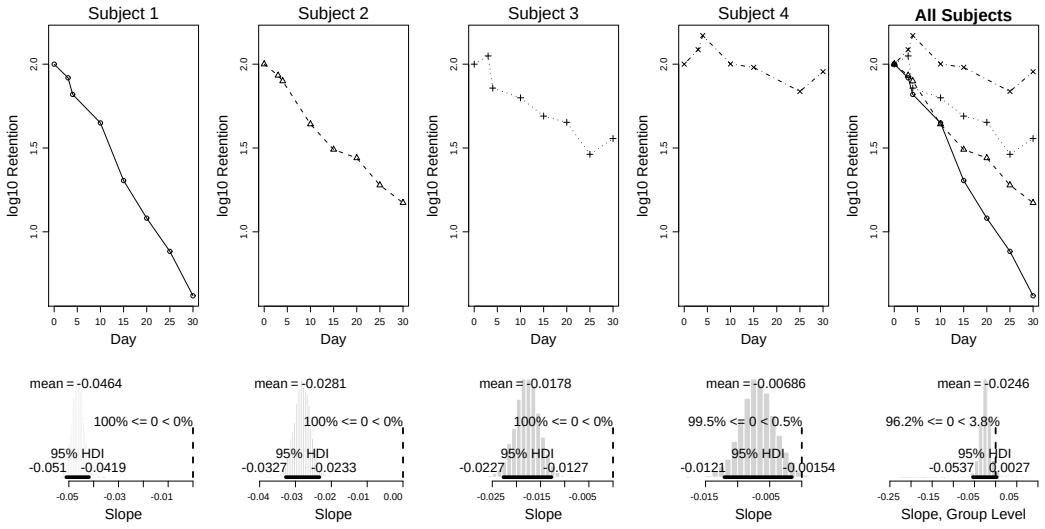

Figure 16.12: For exercise 16.1. Contaminant-retention data with greater variation of individual slopes. Notice the posterior on the group-level slope parameter. Compare with Figure 16.10.

```
for ( subjIdx in 1:4 )  {
rowIdx = ( dataMat[,"subjID"] == subjIdx )
dataMat[rowIdx,"retention"] = dataMat[rowIdx,"retenti  on"] * 10^(subjIdx-1)
}
```

Run the program and include the plot of the data. (See Figure 16.1.) The data curves for the four subjects should have four very different intercepts.

(B) Relative to the original data, are the estimates of the individual slopes different? Relative to the original data, is the posterior mean of the estimated group slope different? Include the histogram of the posterior of the group slope. Is the group slope believably different from zero (according to the 95% HDI)? Why is the group-level slope less believably different from zero, compared to the original data? (Hint: The individual intercepts affect the certainty of the group-average intercept. The group-average intercept trades off with the group-average slope; consider scatterplots of a and mu1c.)

Exercise 16.3. [Purpose: Real data for repeated measures of individual regression, with an outlying individual and non-linear trend.] Suppose we are interested in whether families with more members have higher incomes. The U.S. Census Bureau has published data that indicate the median family income as a function of number of persons in the family, for all 50 states and the District of Columbia and Puerto Rico. The data are plotted in Figure 16.14, p. 369.

(A) Run the program of Section 16.5.3 (SimpleLinearRegressionRepeatedBrugs.R) with the income data. Examine the data section of the program and you will find that the necessary lines of code are already available. The program should generate a figure much like Figure 16.14, p. 369.

(B) There is suggestion of outliers in these data. One curve (for Puerto Rico), falls barely above 20,000, which is far lower than all the others. This suggests a outlier for the distribution of intercepts. Some single data points fall far from the individual linear trends. For example, the 6-person family in the District of Columbia has an income of only about

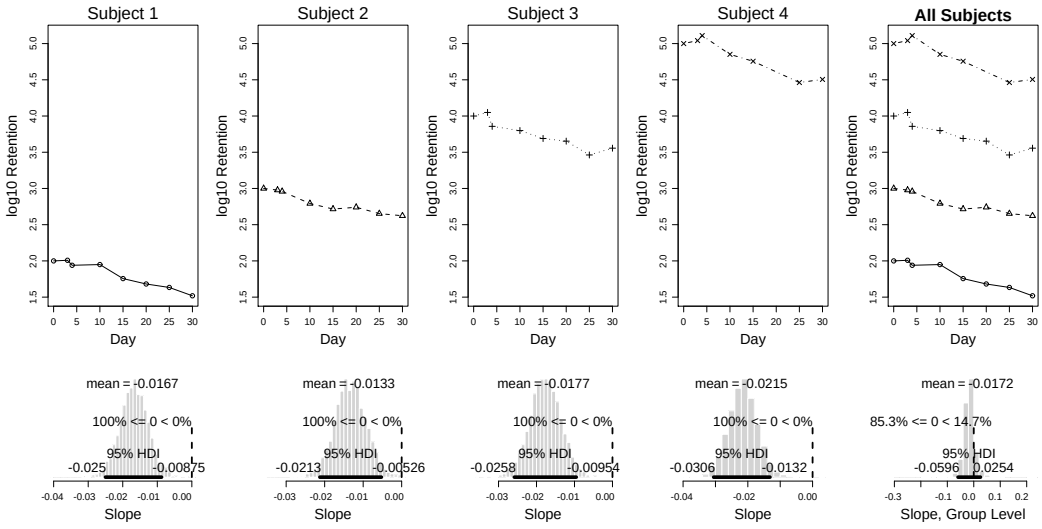

Figure 16.13: For exercise 16.2. Contaminant-retention data with greater variation of individual intercepts. Notice the posterior on the group-level slope parameter. Compare with Figure 16.10.

30,000, whereas the 5-person family has an income of over 60,000. This suggests an outlier for points around linear trends. Finally, some individual slopes seem might be quite steep compared to others. For example, the income in Hawaii rises at $50,000 from 2-person to 7-person families. This increase might or might not be unusual relative to other states. Which distributions in the hierarchical model of Figure 16.1 should be changed to t distributions to address these outliers? Change the model specification to accommodate these t distributions. Use the example in Section 16.5.2 (SimpleRobustLinearRegressionBrugs.R) as a guide. Report your model specification in your write-up. Show the posterior estimate of the intercept of Puerto Rico for small-df t bias and for large-df t bias. Show the posterior estimate of the slope of Hawaii for small-df t bias and for large-df t bias.

(C) The data also suggest a non-linear trend in the data. Income appears to rise for 2, 3, and 4-person families, but then level and decline as family size gets larger. Include in the original (non-t) model another term that can capture "quadratic curvature" in the income level: $= _0 + _1x + _2x^2$. The prior on $_2$ is analogous to the prior on $_1$. Is the group-average estimate of curvature credibly non-zero?

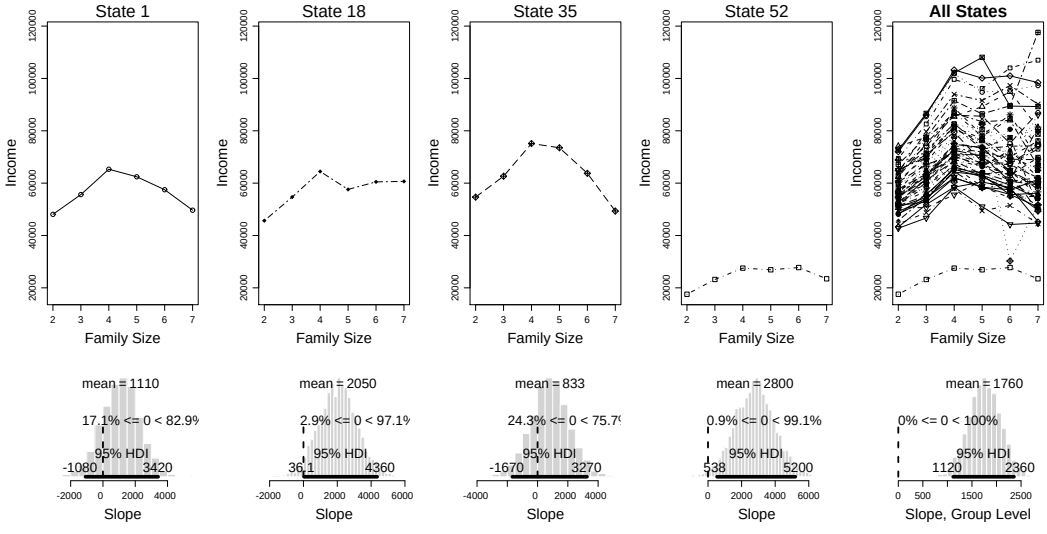

Figure 16.14: For exercise 16.3. The top right graph shows data from 50 states plus Puerto Rico and the District of Columbia, and the first four plots show data from four randomly selected individual states. Lower row shows marginal posterior distribution on the slope parameters.

Chapter 17

Metric Predicted Variable with Multiple Metric Predictors

Contents

> When I was young two plus two equaled four, but
> Since I met you things don't add up no more.
> My keel was even before I was kissed, but
> Now my predictions all come with a twist.

In this chapter we are concerned with situations such as predicting a person's college grade point average (GPA) from his or her high school GPA and scholastic aptitude test (SAT) score. Another such situation is predicting a person's blood pressure from his or her height and weight. In these situations, the value to be predicted is on a metric scale, and there is more than one predictor, each of which is also on a metric scale.

We will consider models in which the predicted variable is an additive combination of predictors, all of which have proportional influence on the prediction. This kind of model is called "multiple linear regression", and is listed in Table 14.1 in its first row and third

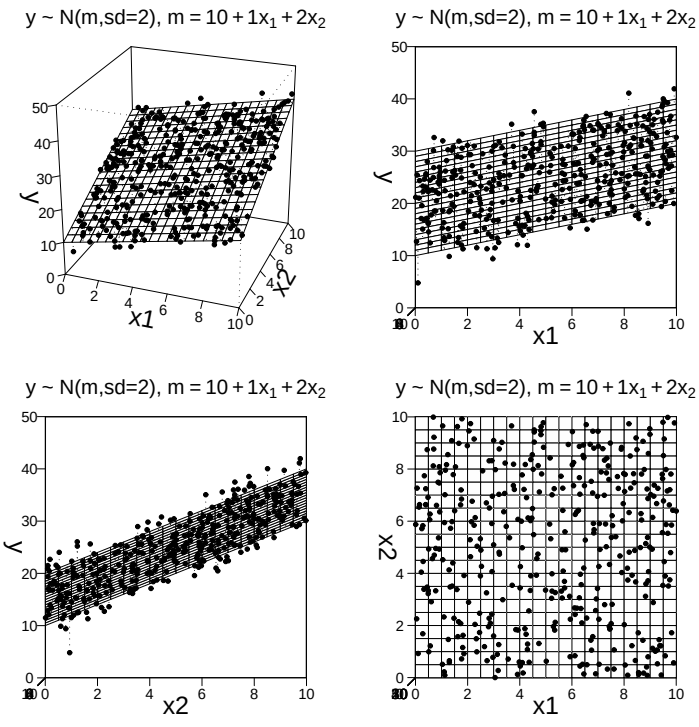

Figure 17.1: Data y, that are normally distributed around the values in the plane. The x_1, x_2 values are sampled uniformly and independently of each other, as shown in the lower-right panel. The panels show different perspectives on the same plane and data. Notice that when the data are plotted against x_1 (marginalized across x_2), the points appear to rise along with the linear function that generated them. Compare with Figure 17.2.

column. We will also consider non-additive combinations of predictors, which are called "interactions".

17.1 Multiple linear regression

Figures 17.1 and 17.2 shows examples of data generated by a model for multiple linear regression. The model specifies the dependency of y on x_1, x_2, but does not specify the distribution of x_1, x_2. At any position x_1, x_2, the values of y are normally distributed in a vertical direction, centered on the height of the plane at that position. The height of the plane is a linear combination of the x_1, x_2 values. Formally, $y \sim N(\mu, \sigma)$, and $\mu = \beta_0 + \beta_1 x_1 + \beta_2 x_2$. For a review of how to interpret the coefficients, β_0, β_1, and β_2, see Figure 14.2, p. 297. The model assumes homogeneity of variance: At all values of x_1 or x_2, the variance of y is the same.

17.1.1 The perils of correlated predictors

Figures 17.1 and 17.2 show data generated from the same model. All that differs between them is the distribution of x_1, x_2, which is not specified by the model. In Figure 17.1, the x_1, x_2 values are distributed uniformly. In Figure 17.2, the x_1, x_2 values are negatively

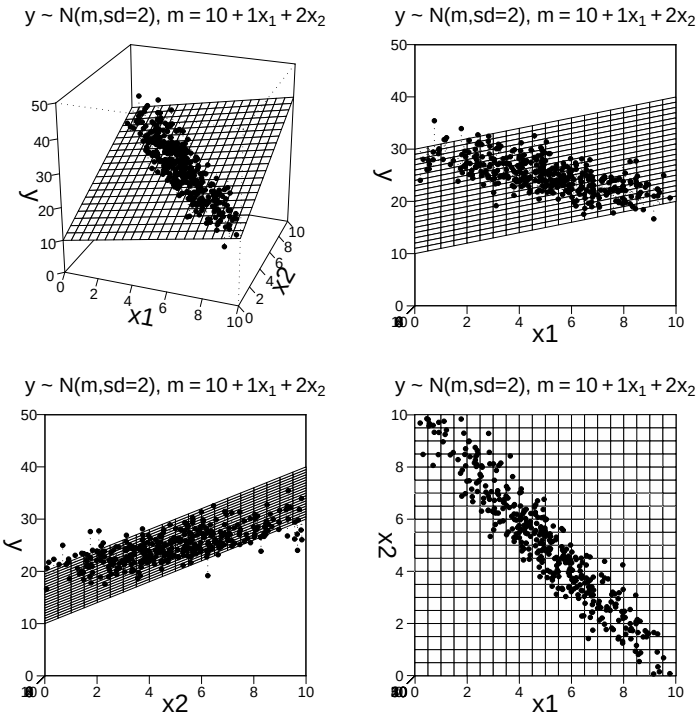

Figure 17.2: Data y, that are normally distributed around the values in the plane. The x_1, x_2 values are correlated, as shown in the lower-right panel. The panels show different perspectives on the same plane and data. Notice that when the data are plotted against x_1 (marginalized across x_2), the points appear to drop, contrary to the linear function that generated them. Compare with Figure 17.1.

correlated: When x_1 is small, x_2 tends to be large, and when x_1 is large, x_2 tends to be small. The correlation of x_1 and x_2 can lead to misinterpretations of their individual influences on y. For instance, notice in Figure 17.2 that when x_1 is near zero, then the data y values are near 30, but when x_1 is near 10, then the data y values are near 20. This observation that y declines from 30 to 20 might leave the impression that an increase in x_1 predicts a decrease in y. But such an impression is wrong, because the data were generated by a function that increases y as x_1 increases; i.e., the coefficient β_1 on x_1 is +1. The reason that the y values appear to decline as x_1 increases is that x_2 decreases when x_1 decreases, and x_2 has an even bigger influence on y than x_1 does.

It is not unusual for predictors to be correlated in real data. For example, consider trying to predict a state's average high school SAT scores on the basis of the amount of money the state spends per pupil. If you plot only mean SAT against mean spent, there is actually a decreasing trend, as can be seen in the lower left panel of Figure 17.3 (data from Guber, 1999). In other words, SAT scores tend to go down as spending goes up. Guber (1999) explains how some political commentators have used this sort of evidence to argue against funding public education.

This negative influence of spending on SAT scores seems quite counter-intuitive. It turns out that the trend is an illusion caused by the influence of another factor, along with the correlation of spending with that other factor. The other factor is the proportion of students who take the SAT. Not all students at a high school take the SAT, because the test

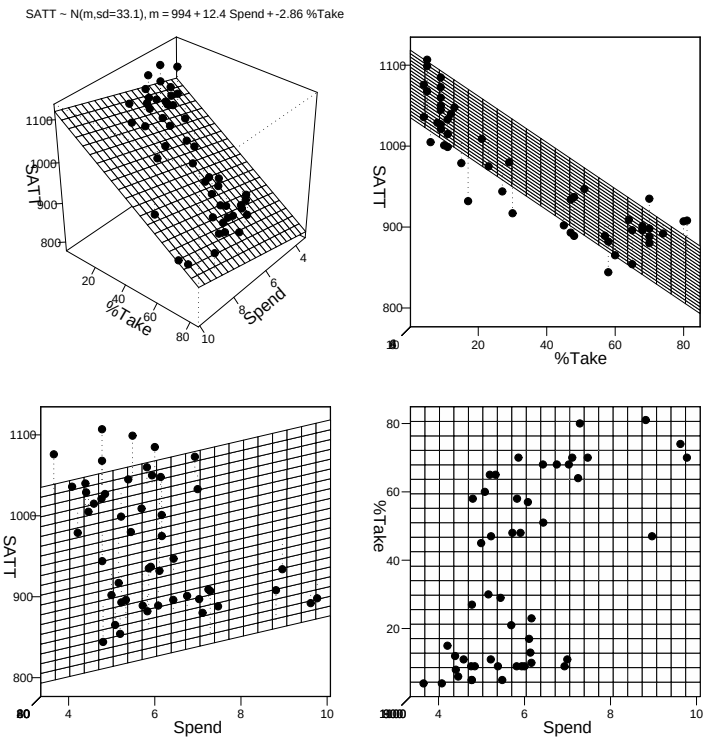

Figure 17.3: An example of multiple linear regression with real data. The data are plotted as dots, and the gridded plane shows the mean of the posterior slopes and intercept. The four panels show different perspectives on the same data and plane. "SATT" is the average total SAT score in a state. "%Take" is the percentage of students in the state who take the SAT. "Spend" is the spending per pupil, in thousands of dollars.

is used primarily for college entrance applications, and therefore it is primarily students who intend to apply to college who take the SAT. Most of the top students at a high school will take the SAT, because most of the top students will apply to college. But students who are weaker academically may be less likely to take the SAT because they are less likely to apply to college. Therefore, the more that a high school encourages mediocre students to take the SAT, lower will be its average SAT score. It turns out that high schools that spend more money per pupil also have a much higher proportion of students who take the SAT. This correlation can be in the lower-right panel of Figure 17.3.

When both predictors, i.e., spending per pupil and percent of students taking the SAT, are taken into account, the influence of spending on SAT scores is seen to be positive, not negative. This positive influence of spending can be seen in the positive slope of the plane along the "Spend" direction in Figure 17.3. The negative influence, of percentage of students taking the SAT, is also clearly shown. To reiterate the main point of this example: It seems that the apparent drop in SAT due to spending is because of spending being correlated with the percentage of students taking the SAT, with the latter having a whoppingly negative influence on SAT scores.

The separate influences of the two predictors could be assessed in this example because the predictors had only mild correlation with each other. There was enough independent variation of the two predictors that their distinct relationships to the outcome variable could

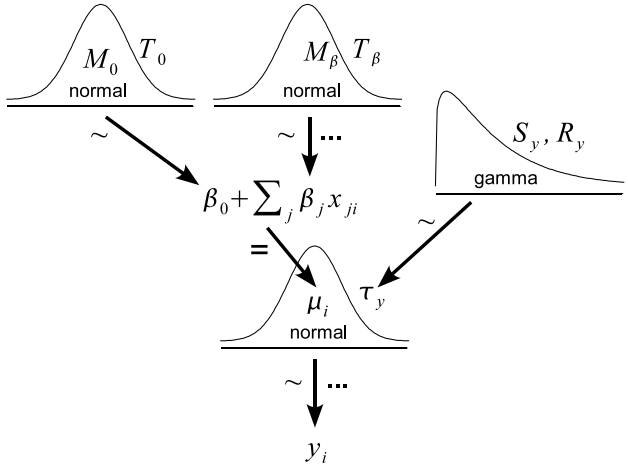

Figure 17.4: Hierarchical diagram for multiple linear regression. (In general, the slope parameters, β_j, may have different priors, instead of the same prior repeated for every j as is shown here for simplicity.)

be detected. In some situations, however, the predictors are strongly correlated and their distinct effects are difficult to tease apart. Correlation of predictors causes the estimates of their regression coefficients to trade off, as we will see when examine the model and its posterior estimates.

17.1.2 The model and BUGS program

The hierarchical diagram for multiple linear regression is shown in Figure 17.4. It is merely a direct expansion of the one for simple linear regression (which appeared in Figure 16.3, p. 346). Instead of just one slope coefficient for a single predictor, there is another slope coefficient for every one of the multiple regressors. For every coefficient, the prior is normal, just as shown in Figure 16.3.

As usual, the BUGS model specification has a line of code for every variable in the hierarchical diagram. The model specification looks like this: (MultipleLinearRegressionBrugs.R)

```
11 model {
12     for( i in 1 : nData ) {
13         y[i] ~ dnorm( mu[i] , tau )
14         mu[i] <- b0 + inprod( b[] , x[i,] )
15     }
16     tau ~ dgamma(.01,.01)
17     b0 ~ dnorm(0,1.0E-12)
18     for ( j in 1:nPredictors ) {
19         b[j] ~ dnorm(0,1.0E-12)
20     }
21 }
```

Notice that what is written in mathematical notation as $\mu_i = \beta_0 + \sum_j \beta_j x_{ij}$ is expressed in BUGS on line 14 by using the BUGS `inprod` function. The `inprod` function is named for the mathematical "inner product" of two vectors. The function takes two vectors and returns the sum of their component-by-component products. Thus, for two vectors, v and w, the inner product `inprod(v[1:n],w[1:n])` is $\sum_{j=1}^{j=n} v_j w_j$. In BUGS, we do not need

to specify the range of indices for the vector when it is on the right side of an assignment operator, and we intend to use all components of the vector. But in BUGS we do need to include the square brackets so that BUGS knows that the variable is a vector. If we wanted to be completely explicit, we could write the inner product in line 14 as `inprod( b[1:nPredictors] , x[i,1:nPredictors] )`. The complete program is listed in Section 17.5.1 (MultipleLinearRegressionBrugs.R).

The BUGS model specification happens to put the same prior on every slope parameter. This equivalence is applied as a generic convenience, but is not required. Indeed, if there is prior information that suggests different priors on different predictors, then the prior knowledge should be respected.

17.1.2.1 MCMC efficiency: Standardizing and initializing

As described previously in Section 16.1.1.1 (p. 347), the MCMC sampling can be made much more efficient if the data are standardized. Standardizing each variable is straightforward. The MCMC sampling then finds regression coefficients that are appropriate for the standardized data. We would like to transform the parameters to the corresponding values that are appropriate to the original, nonstandardized scores. This can be done by generalizing Equation 16.2 to multiple predictors:

$$z_{\hat{y}} = \beta_0 + \sum_j \beta_j z_{x_j}$$

$$\frac{\hat{y} - M_y}{SD_y} = \beta_0 + \sum_j \beta_j \frac{x_j - M_{x_j}}{SD_{x_j}}$$

$$\hat{y} = \underbrace{\beta_0 SD_y + M_y - \sum_j \beta_j SD_y M_{x_j} / SD_{x_j}}_{\beta_0} + \sum_j \underbrace{\beta_j SD_y / SD_{x_j}}_{\beta_j} x_j \qquad (17.1)$$

The estimate of β_y is merely $z_{\beta_y} SD_y$.

Even after standardizing, it can also help to start the chains near their posterior credible values. To do this, we use the built in linear model function in R, called `lm`. There is no need to delve into the inner workings of `lm`, but suffice it to say that it returns the maximum-likelihood estimate (MLE) of the intercept and slope coefficients, in raw scales. These raw-scale slope coefficients can be easily converted to standardized scales, and to initialize the chains. In conclusion, the standardization of the data makes the chain efficient and less autocorrelated once it reaches the modal region of the posterior, and the initialization at the MLE implies that the burn-in period is minimal.

17.1.3 The posterior: How big are the slopes?

Figure 17.5 shows the posterior distribution the results of the SAT data in Figure 17.3 and model in Figure 17.4. You can see that the slope on Spend is credibly well above zero, with a mean slope of about 12.3, which suggests that SAT scores rise by about 12.3 points for every extra $1,000 spent per pupil. The slope on Percent taking the SAT is also credibly non-zero, with a mean of -2.85, which suggests that SAT scores fall by about 2.85 points for every additional 1% of students who take the test.

The scatter plots in the bottom of Figure 17.5 show correlations among the credible parameter values. In particular, the lower-right scatter plot shows that the coefficient for

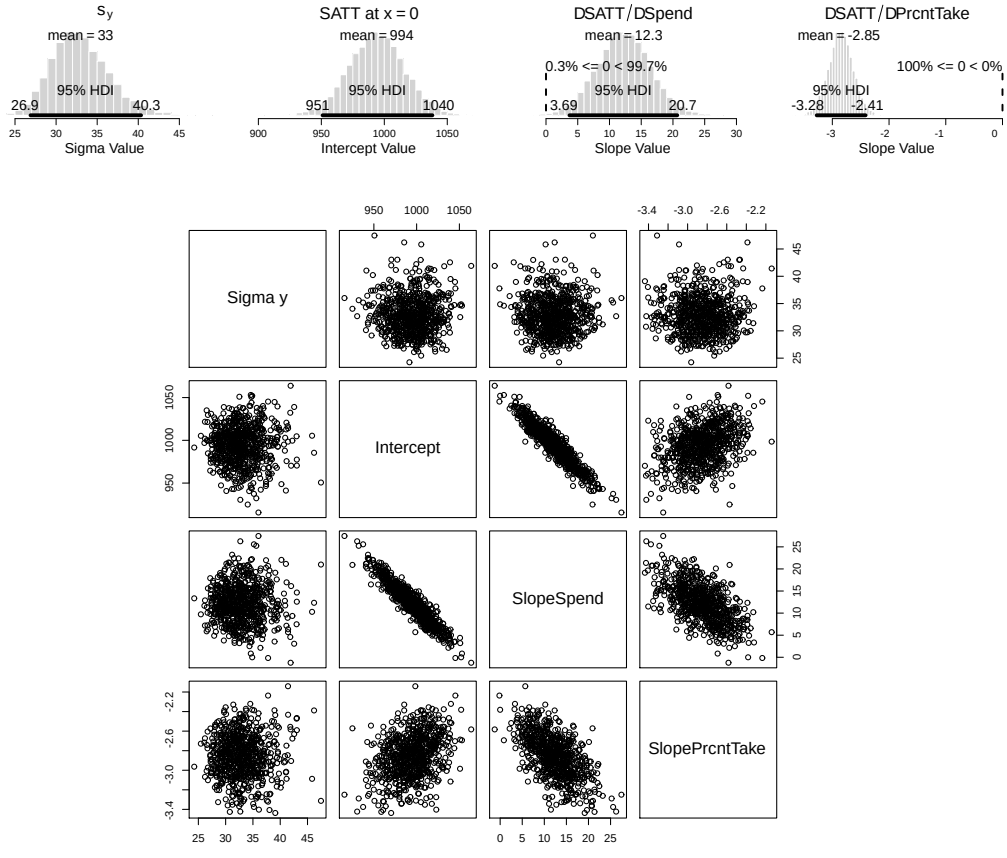

Figure 17.5: Posterior distribution for data in Figure 17.3 and model in Figure 17.4. Upper row indicates HDI's on regression coefficients. Scatterplots reveal correlations among credible parameter values; in particular, the coefficient on Spending ("SlopeSpend") trades off with the coefficient on Percentage taking the SAT ("SlopePrcntTake"), because those predictors are correlated in the data.

Spending trades off with the coefficient on Percentage taking the SAT. The correlation means that if we believe that the influence of Spending is large, then we must believe that the influence of Percentage Taking is smaller, to stay consistent with the data. Conversely, if we believe that the influence of Spending is small, then we must believe that the influence of Percentage Taking is larger. This makes sense because those two predictors are correlated in the data, and therefore the two predictors are not differentially constraining the regression coefficients. Think of this simple example for two data points: Suppose $y = 1$ for $x_1 = 1$ and $x_2 = 1$, and $y = 2$ for $x_1 = 2$ and $x_2 = 2$. The linear model $y = \beta_1 x_1 + \beta_2 x_2$, is supposed to satisfy both data points, which implies that $1 = \beta_1 + \beta_2$. Therefore, to satisfy the data, it could be that $\beta_1 = 2$ and $\beta_2 = -1$, or $\beta_1 = 0.5$ and $\beta_2 = 0.5$, or $\beta_1 = 0$ and $\beta_2 = 1$, etc. In other words, the credible values of β_1 and β_2 are (anti-)correlated.

One of the benefits of Bayesian analysis is that correlations of credible parameter values are explicit in the posterior distribution. Traditional statistical methods provide only a single "best" (e.g., MLE) parameter value, without indicating the trade-offs among parameter values. The Bayesian posterior, however, naturally shows the tradeoffs and redundancies among parameters. It is up to the user to actually look for and interpret the correlations of

parameters, of course. Another bene t of Bayesian analysis is that the model doesn't "explode" when predictors are correlated. If predictors are correlated, the joint uncertainty in the regression coe cients is evident in the posterior, but the model happily generates a posterior regardless of correlations in the predictors. Traditional, one-best-solution method is much less robust in the presence of strongly correlated predictors.

17.1.4 Posterior prediction

Often we are interested in using the linear model to predict y values for various x values. It is straight forward to generate a large sample of credible y values for speci ed x values. From the distribution of y values we can compute the mean and HDI to summarize the centrally predicted y value and the uncertainty of the prediction. As was the case in simple linear regression, illustrated back in Figure 16.6, the uncertainty in predicted y is greater for x values outside the bulk of the data. In other words, extrapolation is more uncertain than interpolation.

The last part of the code in Section 17.5.1 (MultipleLinearRegressionBrugs.R) carries out these computations. At every step in the MCMC chain of credible parameter values, the program randomly generates a y value based on the linear model. The x values, for which posterior predictions are desired, are stored in the matrix `xPostPred`, one row per point to be predicted. Thus `xPostPred` has as many columns as there are predictors. The matrix `yPostPred` also has one row per point to be predicted, with each row containing randomly generated y values for the corresponding point. The program generates one y value per step in the MCMC chain, hence `yPostPred` has as many columns as the are steps in the chain. The vector `b0Samp` contains posterior values of the intercept, β_0, at each step in the MCMC chain. The vector `bSamp[chainIdx,]` contains the slope coe cients, β_j, for the predictors, at one step in the chain. The slope coe cients are forced to be a column vector by passing them through the `cbind` function in R. Then the slopes can be multiplied by the corresponding x values and summed together, simultaneously for all the predictors, in one matrix operation: `xPostPred %*% cbind(bSamp[chainIdx,])`. This matrix operation appears on line 235 in the following code: (MultipleLinearRegressionBrugs.R)

```
232 for ( chainIdx in 1:chainLength ) {
233     yPostPred[,chainIdx] = rnorm( NROW(xPostPred) ,
234                                   mean = b0Samp[chainIdx]
235                                          + xPostPred %*% cbind(bSamp[chainIdx,]) ,
236                                   sd = rep( sigmaSamp[chainIdx] , NROW(xPostPred) ) )
237 }
```

Notice that the code loops through every step in the MCMC chain, filling in the `yPostPred` matrix one column at a time. The values are generated randomly from a normal distribution, using the `rnorm` function, with a standard deviation of `sigmaSamp[chainIdx]`.

17.2 Hyperpriors and shrinkage of regression coe cients

In some research, there are many candidate predictors which we suspect could possibly be informative about the predicted variable. For example, when predicting college GPA, we might include high school SAT, high school GPA, income of student, income of parents, years of education of the parents, spending per pupil at student's high school, student IQ, student height, weight, shoe size, hours of sleep per night, distance from home

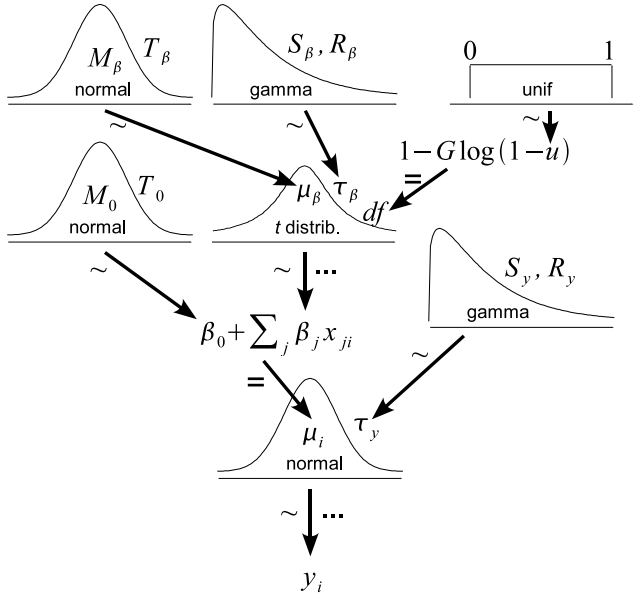

Figure 17.6: Hierarchical diagram for multiple linear regression, with hyperprior on slope coefficients across predictors. Compare with Figure 17.4.

to school, amount of caffeine consumed, hours spent studying, hours spent earning age, weight, etc. We can include all the candidate predictors in the model, with a regression coefficient for every predictor. Should all those regression coefficients be estimated in isolation from the others, as in the model of Figure 17.4? Probably not, because we probably have some prior knowledge that relates the influences of the predictors to each other. If nothing else, we can at least say that the candidate predictors all come from the set of remotely plausible predictors. What do we know about this set of remotely plausible predictors? Most candidate predictors probably have a very small relationship with the predicted variable, but a few candidate predictors may have sizable covariation with the predicted variable. In other words, the regression coefficients are probably distributed something like a t distribution, with lots near a mean of zero, but with a few outliers in the extended tails. We therefore put this prior knowledge into the model structure, as shown in Figure 17.6. (This method is mentioned in passing by Gelman et al., 2004, p. 405)

Figure 17.6 indicates that the regression coefficients, β_j, are distributed as a t distribution. The parameters of the t distribution are estimated from the data. Presumably, many of the credible regression coefficients will be near zero, but a few will depart a lot from zero, and the t distribution will have credible μ and df values that reflect the distribution of regression coefficients in the data. Exercise 17.1 has you generate the prior in BUGS, so you get a better intuition for how the constants in the hyperprior distribution affect the implied prior on the regression coefficients.

A desirable side-effect of incorporating this prior structure is that the estimates of the regression coefficients experience shrinkage. If many regression coefficients are near zero, then the t distribution will have a high precision (parameter), which in turn will shrink the estimates of the regression coefficients. The regression coefficients are mutually informing each other, via the prior knowledge that they should be distributed according to a t distribution.

The shrinkage is desirable not only because it expresses prior knowledge, but also

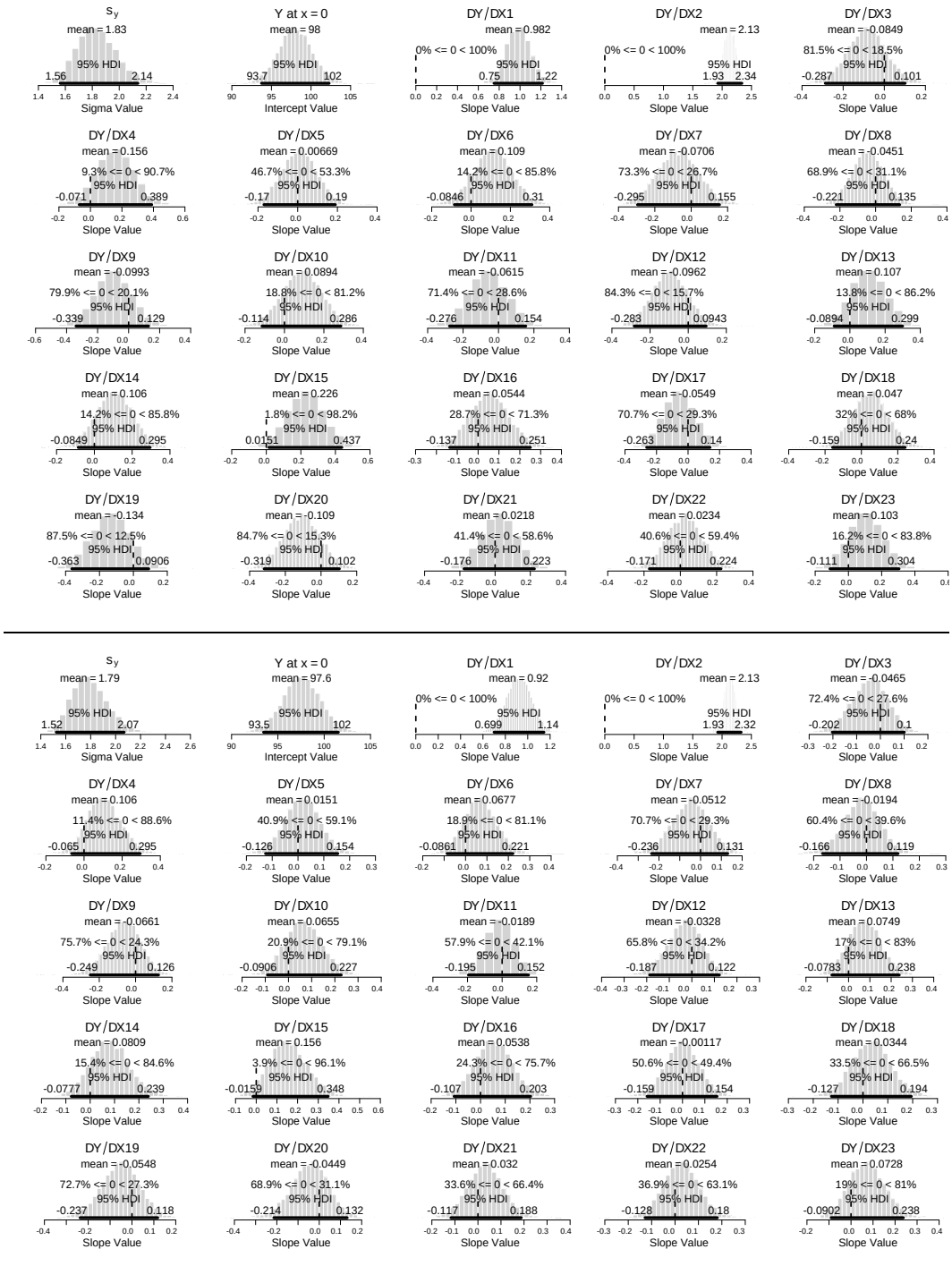

Figure 17.7: Posterior distribution for 100 data points randomly generated with true parameter values of $_y = 2$, $_0 = 100$, $_1 = 1$, $_2 = 2$, and all other $_j = 0$. Upper half: Model is from Figure 17.4, wherein each slope coefficient is insulated from the other. Notice the "false alarm" for $_{15}$. Lower half: The model is shown in Figure 17.6, wherein the slope coefficients mutually inform each other. Shrinkage prevents the "false alarm" for $_{15}$.

because it rationally helps control for "false alarms" in claiming that a predictor has a non-zero regression coefficient. When there are many candidate predictors, some of them may spuriously appear to have credibly non-zero regression coefficients, even when the true coefficient is zero. This sort of false alarm is unavoidable because data are randomly sampled, and there will be occasional coincidences of data that are unrepresentative. By letting the regression coefficients be mutually informed by other predictors, and not only by the data of the single predictor each multiplies, the coefficients are less likely to be spuriously distorted by a rogue sample.

As an example in which we can specify the true regression coefficients, let's randomly generate 100 data points from a linear regression model that has parameter values of $_y = 2$, $_0 = 100$, $_1 = 1$, $_2 = 2$, and all other $_j = 0$ for 21 other predictors. When using insulated regression coefficients for each predictor, as in the hierarchical diagram of Figure 17.4, the resulting posterior estimates are shown in the upper half of Figure 17.7. Notice that the estimate of $_{15}$ (denoted Y= X15) suggests that this predictor may have a non-zero regression coefficient. (If we could specify a ROPE of some non-zero width, we might decide that $_{15}$ is non-zero, but ROPEs are best defined in meaningful contexts, not in a generic example like this.) In other words, this apparent non-zero value of $_{15}$ is a "false alarm", produced by quirks in the random sample of data.

The estimates in the upper half of Figure 17.7 did not use any knowledge about how the regression coefficients might be related. If we instead use the model of Figure 17.6, with the same data, the resulting posterior is shown in the lower half of Figure 17.7. Notice that the estimate of $_{15}$ now includes zero inside the HDI. All the estimates of the regression coefficients are reduced a little, relative to the upper half of the figure. The reason for this shrinkage is that the many regression coefficients near zero mutually inform each other via the overarching t distribution.

When using the model of Figure 17.6, the shrinkage of the regression coefficients is toward their mean, , not necessarily toward zero. If your prior knowledge suggests that the true mean of the regression coefficients is very close to zero, then that knowledge should be expressed in the model of Figure 17.6 by setting the mean to zero and setting the precision $_T$ to a very large value such as 10,000. This setting implies that is already fairly certain, and the real prior uncertainty is in the variability of regression coefficients across predictors.

The hyperprior over regression coefficients, shown in Figure 17.6, is meant to express a genuine belief, namely, that all the regression coefficients can be reasonably described by a t distribution. This assumption may be more or less tenable in different situations. For example, a region's agricultural crop yield might be predicted from rainfall amounts measured at 39 randomly selected locations. The predictors from each of these measuring locations are probably fairly similar, except for perhaps a few outliers, and therefore it seems quite reasonable to model the regression coefficients by a t distribution. In other situations, it may be less reasonable to treat all the regression coefficients as coming from a shared t distribution. At the least, the predictors were selected from some implicit set of reasonably likely predictors, and we can think of the overarching distribution as reflecting that set. We might still use the hyperprior model, but only as a convenience to impose some degree of shrinkage on the regression coefficients. It should be interpreted carefully. Beware of convenience priors that are used in routinized fashion.

17.2.1 Informative priors, sparse data, and correlated predictors

This book has emphasized the use of mildly informed priors, as opposed to conventionalized noninformed priors or strongly informed priors. On the other hand, this book has also mentioned that a benefit of Bayesian analysis is the potential for cumulative scientific progress through the use of priors that have been informed by previous research. Informed priors should be used whenever the skeptical scientific audience of the analysis deems it appropriate, especially if the analysis is accompanied by a check of posterior robustness. If the conclusion from the posterior is strong, even with an uninformed prior, then the mildly informed prior should be used, because a broader audience may find the analysis to be convincing. But there are situations in which the use of an informed prior is appropriate.

In general, when the data are sparse (i.e., when the sample size is small), the posterior will be imprecise if the prior is imprecise. But if the prior is not weakly constrained, then a small amount of data can lead to a more decisive posterior. An example of this phenomenon appeared in Exercise 5.7, p. 81, in which the prior allowed only two very different interpretations of the data. In this case, even a small amount of data shifted the posterior toward one or the other interpretation.

Sparse data can also lead to usefully precise posteriors in the context of multiple linear regression, if some of the regression coefficients use informed priors, and the predictors are correlated. To develop this idea, it is important to remember that when predictors are correlated, their regression coefficients are also (anti-)correlated. For example, recall the SAT data from Figure 17.3, p. 374, in which the predictors are correlated in the data, i.e., spending-per-pupil and percent-taking-the-exam are related. Consequently, the posterior estimates of the regression coefficients had a negative correlation, as shown in Figure 17.5, p. 377. The correlation of credible regression coefficients implies that a strong belief about the value of one regression coefficient constrains the value of the other coefficient. Look carefully at the scatterplot of the two slopes shown in Figure 17.5. It can be seen that if we believe that the slope on percent-taking-the-exam is 2, then credible values of the slope on spending-per-pupil are around 18, with an HDI of roughly 13 to 22. Notice that this HDI is much smaller than the marginal HDI on spending-per-pupil, which goes from 3.7 to 20.7. Thus, constraining the beliefs about one slope also constrains beliefs about the other slope, because estimates of the slopes are correlated.

That influence of one slope estimate on another can be used to differential advantage when we have prior knowledge about one of the slopes. If some previous or auxiliary research informs the prior of one regression coefficient, that constraint can propagate to the estimates of regression coefficients on other predictors that are correlated with the first. This is especially useful when the sample size is small, and therefore mildly informed prior would not yield a very precise posterior. Of course, the informed prior on the first coefficient must be cogently justified for the skeptical audience. A robustness check also may be useful, to show how strong the prior must be to drive strong conclusions. If the information used for the prior is compelling, then this technique can be very useful for leveraging novel implications from small samples. An accessible discussion and example from political science is provided by Western and Jackman (1994), and a mathematical discussion is provided by Learner (1978, p. 175).

17.3 Multiplicative interaction of metric predictors

In some situations, the predicted value is not merely an additive combination of the predictors. For example, the effects of drugs are often non-additive. As the dosage of one drug increases, there might be a moderate increase in effectiveness. And as the dosage of another drug increases (when administered by itself), there may be a moderate increase in effectiveness. But when the two drugs are administered together, when the dosages of both drugs are high, they might interact to greatly enhance or greatly decrease effectiveness, beyond a mere addition of the two separate effects. As another example, consider trying to predict subjective happiness from income and health. If either income or health is low, subjective happiness is probably also fairly low. But if both income and health are high, then happiness is more likely to be high. In other words, happiness might not increase additively with both income and health; instead, it happiness may be enhanced by an interaction of income and health. In general, interaction means that the influence of one predictor varies, depending on the value of the other predictor.

Formally, interaction can be expressed in different ways. We will consider multiplicative interaction. For two metric predictors, regression with multiplicative interaction has these algebraically-equivalent expressions:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{1\,2} x_1 x_2 \tag{17.2}$$

$$= \beta_0 + \underbrace{\beta_1 + \beta_{1\,2} x_2}_{\text{slope of } x_1} x_1 + \beta_2 x_2 \tag{17.3}$$

$$= \beta_0 + \beta_1 x_1 + \underbrace{\beta_2 + \beta_{1\,2} x_1}_{\text{slope of } x_2} x_2 \tag{17.4}$$

These three expressions emphasize different interpretations of interaction, as illustrated in Figure 17.8.

The form of Equation 17.2 is illustrated in the left panel of Figure 17.8. The vertical arrows show that the curved-surface interaction is created by adding the product, $\beta_{1\,2} x_1 x_2$, to the planar linear combination.

The form of Equation 17.3 is illustrated in the middle panel of Figure 17.8. Its dark lines show that the slope in the x_1 direction depends on the value of x_2. In particular, when $x_2 = 0$, then the slope along x_1 is β_1, which in the graphed example is 1. But when $x_2 = 10$, then the slope along x_1 is $\beta_1 + \beta_{1\,2} x_2$, which in the graphed example is 1. Again, the slope in the x_1 direction changes when x_2 changes, and β_1 only indicates the slope along x_1 when $x_2 = 0$.

The form of Equation 17.4 is illustrated in the right panel of Figure 17.8. It shows that the interaction can be expressed as the slope in the x_2 direction changing when x_1 changes. This is exactly analogous to the middle panel of Figure 17.8, but it is important to realize, and visualize, that the interaction can be expressed in terms of the slopes on either predictor.

Great care must be taken when interpreting the coefficients of a model that includes interaction terms (Braumoeller, 2004). In particular, lower-order terms are especially difficult to interpret when higher-order interactions are present. In the simple two-predictor case, the coefficient β_1 describes the influence of predictor x_1 only at $x_2 = 0$, because the slope on x_1 is $\beta_1 + \beta_{1\,2} x_2$, as was shown in Equation 17.3 and graphed in the middle panel of Figure 17.8. In other words, it is not appropriate to say that β_1 indicates the overall influence of x_1 on y_1. Indeed, in many applications, the value x_2 never realistically gets close to

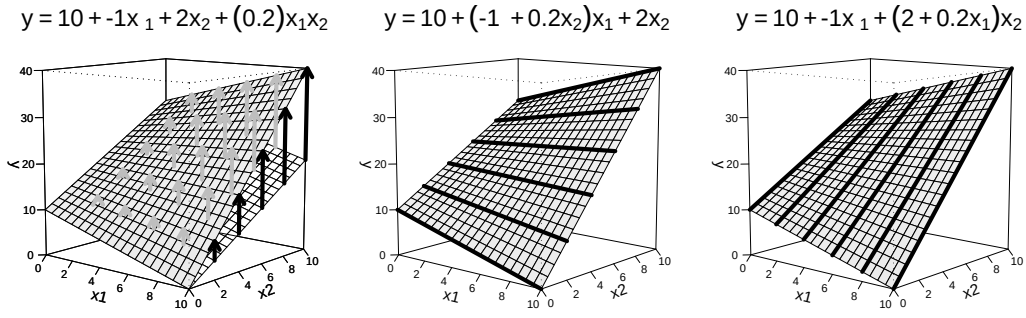

Figure 17.8: A multiplicative interaction of x_1 and x_2, parsed three ways. The left panel emphasizes that the interaction involves a multiplicative component that adds a vertical amount to the planar additive model, as indicated by the arrows. The middle panel shows the same function, but with the terms algebraically re-grouped to emphasize that the slope in the x_1 direction depends on the value of x_2, as shown by the darkened lines. The right panel again shows the same function, but with the terms algebraically re-grouped to emphasize that the slope in the x_2 direction depends on the value of x_1, as shown by the darkened lines.

zero, and therefore β_1 has no realistic interpretation at all. For example, suppose we are predicting college GPA (y) from parental income (x_1) and high school GPA (x_2). If there is interaction, then the regression coefficient, β_1, on parental income, only indicates the slope on x_1 when x_2 (GPA) is zero. Of course, there are no GPAs of zero, and therefore β_1 by itself is not very informative.

17.3.1 The hierarchical model and BUGS code

The model for regression with multiplicative interaction is the same as for linear regression but with an added term for the interaction. Because we are estimating the coefficient on the multiplication of the predictors, that coefficient must have a prior, analogous to the priors on the linear coefficients. For the case of two predictors, with an interaction term, the BUGS model specification looks like this: (MultiLinRegressInteractBugs.R)

```
model {
    for( i in 1 : nData ) {
        y[i] ~ dnorm( mu[i] , tau )
        mu[i] <- b0 + b1 * x[i,1] + b2 * x[i,2] + b12 * x[i,1] * x[i,2]
    }
    tau ~ dgamma(.001,.001)
    b0 ~ dnorm(0,1.0E-12)
    b1 ~ dnorm(0,1.0E-12)
    b2 ~ dnorm(0,1.0E-12)
    b12 ~ dnorm(0,1.0E-12)
}
```

Notice that the interaction coefficient, `b12`, has a normal prior analogous to the priors on the slope coefficients.

In models that involve many predictors, there can be many possible interaction terms. Just as we can put a hyperprior on the slopes, we can also put a hyperprior on the interaction coefficients. The idea is that most interaction terms are near zero, but a few might depart

from zero. This distribution over interaction terms could be modeled as a t distribution. We will not pursue examples that involve numerous interactions.

17.3.1.1 Standardizing the data and initializing the chains

We will, as before, standardize the data before entering them into the BUGS model. This helps reduce correlations in the parameters, but does not eliminate correlations. When initializing the chains, it may suce to set the interaction coecient(s) to zero, and start the slopes at their MLE values for a non-interactive model, especially when the interactions are small, as is often the case.

Transforming the standardized estimates back to the original scales is conceptually simple but algebraically much messier when interaction terms are involved. The expression, when there are merely two predictors, turns into this unwieldy form:

$$
\begin{aligned}
z(y) &= \beta_0 + \beta_1 z(x_1) + \beta_2 z(x_2) + \beta_{1:2} z(x_1)\, z(x_2) \\
\frac{y - m_y}{s_y} &= \beta_0 + \beta_1 \frac{x_1 - m_1}{s_1} + \beta_2 \frac{x_2 - m_2}{s_2} + \beta_{1:2} \frac{x_1 - m_1}{s_1} \frac{x_2 - m_2}{s_2} \\
y &= m_y + s_y \left(\beta_0 - \beta_1 \frac{m_1}{s_1} - \beta_2 \frac{m_2}{s_2} + \beta_{1:2} \frac{m_1 m_2}{s_1 s_2} \right) \\
&\quad + \left(\beta_1 \frac{1}{s_1} - \beta_{1:2} \frac{m_1}{s_1} \frac{1}{s_2} \right) x_1 \\
&\quad + \left(\beta_2 \frac{1}{s_2} - \beta_{1:2} \frac{1}{s_1} \frac{m_2}{s_2} \right) x_2 \\
&\quad + \beta_{1:2} \frac{1}{s_1} \frac{1}{s_2} x_1 x_2
\end{aligned}
\tag{17.5}
$$

When there are more predictors involved, with their interactions, the expression becomes quite protracted. In those situations, there is no avoiding matrix notation and matrix algebra, which greatly facilitates manipulating the forms. We will not venture into matrix expressions for the GLM in this book (although we did splurge a bit on matrices in Section 7.1.5, p. 104). Matrix operations are not easily expressed in BUGS, unfortunately.

17.3.2 Interpreting the posterior

To illustrate some of the issues involved in interpreting the parameters of a model with interaction, consider again the SAT data from Figure 17.2. Recall that the mean SAT score in a state was predicted from the spending per pupil and the percentage of students who took the test. When no interaction term was included in the model, the posterior looked like Figure 17.5, which indicated a positive influence of spending and a negative influence of percentage of students.

Would we want to include an interaction term in the model? The meaning of interaction is that the effect of one predictor changes when the level of the other predictor changes.

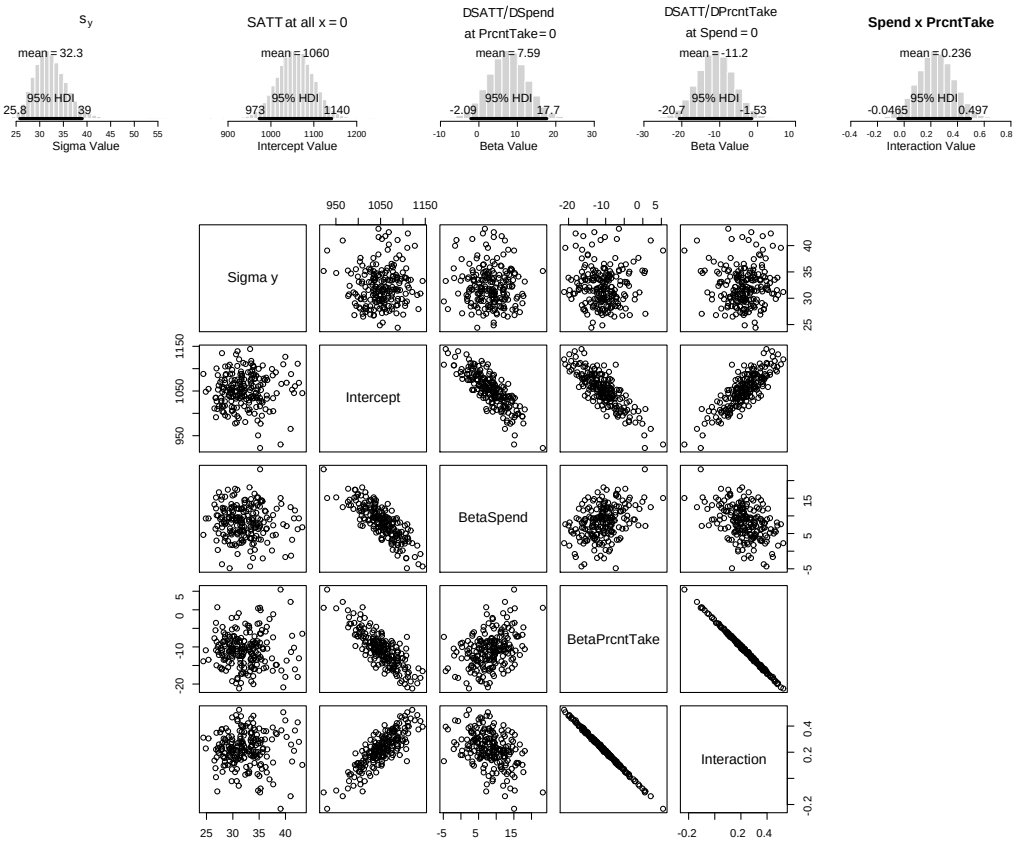

Figure 17.9: Posterior for SAT data in Figure 17.3. Compare this posterior, which includes an interaction term, with the posterior that excludes an interaction, shown in Figure 17.5.

Does it make sense in this application that the effect of spending would depend on the percentage of students taking the test? Perhaps yes, because when very few students are taking the test, they are probably already at the top of the class and therefore might not have as much head-room for increasing their scores. In other words, it is plausible that the effect of spending is larger when the percentage of students taking the test is larger, and we would not be surprised if there were a positive interaction between these predictors. Therefore, it is theoretically meaningful to include an interaction term in the model.

When we incorporate a multiplicative interaction into the model, the posterior looks like Figure 17.9. The top right histogram indicates that the most believable interaction coefficients is indeed positive, as we anticipated it could be. 95% HDI includes zero, however, which indicates that we do not have very strong precision in our estimate of the magnitude of the interaction. The scatterplots in the lower part of Figure 17.9 indicate that the interaction coefficient is very strongly correlated with the beta coefficient on percentage of students taking the test. If the MCMC sampling were taking place in these original scales, it would be very inefficient.

A cursory look at the middle histogram of Figure 17.9 might lead a person, inappropriately, to conclude that there is not a credible influence of spending on SAT scores, because zero is among the credible values of β_{Spend}. This conclusion is inappropriate because β_{Spend}

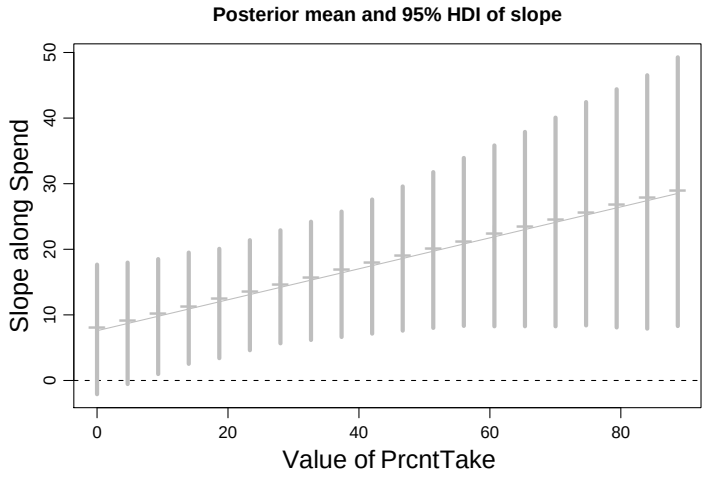

Figure 17.10: The slope in the x_{spend} direction is $_{spend} + _{spend\ \%take} x_{\%take}$. Shown here are the means and 95% HDIs for the slope in the x_{spend} direction, for various values of $x_{\%take}$.

only indicates the slope on spending when the percentage of students taking the test is zero. The slope changes when the percentage of students changes because of the interaction. Because the interaction tends to be positive, the slope of spending increases when the percentage of students taking the test increases.

Figure 17.10 shows the increase in slope along spending as the percentage taking the test increases. Also plotted is the extent of the 95% HDI of the estimated slope. Notice that the HDI at PrcntTake=0 matches the HDI shown in the middle histogram of Figure 17.9.

Interestingly and importantly, the extent of the HDI is not constant, but also depends on the percentage of students taking the test. Mathematically, the change in extent of the HDI stems for two sources. First, the slope along x_1 is $_1 + _{1\ 2} x_2$, which means that the uncertainty in $_{1\ 2}$ is being multiplied by x_2, and therefore the uncertainty in the slope depends on x_2. But the uncertainty in the slope does not necessarily always increase when x_2 increases because $_{spend}$ and $_{spend\ \%take}$ are negatively correlated (see the scatterplot in Figure 17.9): When x_{spend} has a modest size, its negative correlation removes a bit of uncertainty in the slope. This relationship can be seen graphically in Figure 17.11, which shows an idealized subset of credible values for $_1$ and $_{1\ 2}$. The left panel shows that when $x_2 = 0$, the credible parameters (i.e., the dots) span a range of x1 slopes from 0 to 15. The middle panel shows that when $x_2 = 25$, the parameters span a range of x1 slopes from 10 to 15, i.e., a much smaller range. The right panel shows that when $x_2 = 50$, the parameters span a range of x1 slopes from 10 to 25, again a large range. Thus, because of the negative correlation of credible values of $_1$ and $_{1\ 2}$, the narrowest range of x1 slopes is at an intermediate value of x2.

In summary, when there is interaction, then the influence of individual predictors cannot be summarized by their individual regression coefficients alone, because those coefficients only describe the influence when the other variables are at zero. A careful analyst considers credible slopes across a variety of values of the other predictors, as in Figure 17.10. Notice that this is true even when the interaction coefficient did not exclude zero from its 95% HDI: Even though the estimate of the interaction was not very precise,

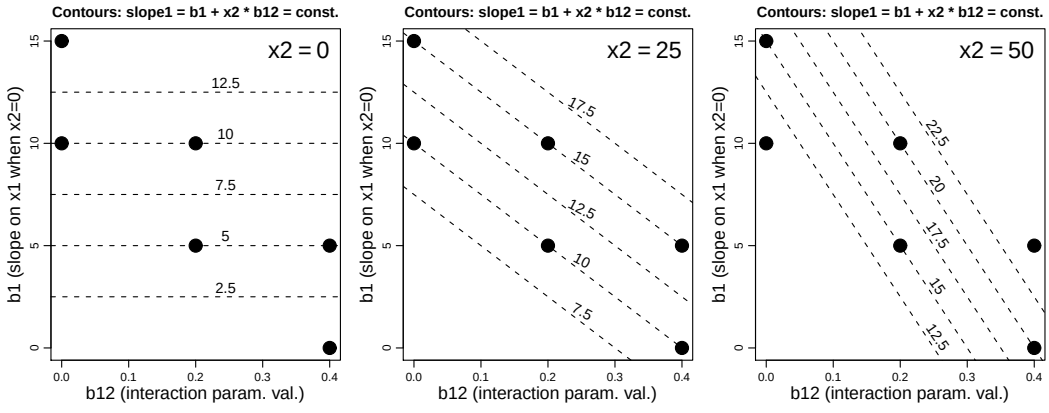

Figure 17.11: Dots show idealized parameter-value combinations, the same in all three panels. Contour lines mark parameter value combinations for slopes in the x1 direction when x2 has that value indicated in the panel.

the interaction did have a considerable influence on the interpretation of the predictors.

17.4 Which predictors should be included?

In many research projects, there are specific theoretical motivations for measuring particular predictors and the predicted variable. In these situations, we know in advance which predictors and interactions we are interested in modeling. But in other situations, we come to a pre-existing database that has many variables measured for other purposes, and we are curious to know which of the variables might be good predictors of a particular variable of interest. Or, we might have some hybrid situation, in which we have particular predictors that we think may be relevant, but we measure several other variables on the chance that they might be relevant, or because they are needed for some other study, or because they are trivial to measure so we go ahead and measure them.

When there are many potential predictors, which ones should be included in our model? And which interactions? A reasonable answer is to include the theoretical predictors and interactions that you think would have any chance of providing useful predictive information. If you do not include them, then you have essentially set their corresponding regression coefficients to zero with complete certainty.

Whether or not to include candidate predictors depends on the purpose of the regression analysis. If the goal is to predict the outcome as accurately as possible, using any predictors at all, regardless of how those predictors might be causally related to the predicted variable, then all reasonable predictors could be included. On the other hand, if the purpose of the analysis is to explain the outcome on the basis of the predictors, then only the predictors that can be meaningfully related to the outcome should be included (see, e.g., Keith, 2005, p. 70).

There are several costs of including a lot of candidate predictors. One cost is unwieldy interpretation of the results. With too many predictors and interactions, the complexity of the mathematical description may provide little meaningful insight into the data. Unless you really believe that the effect of income on happiness depends on shoe size, and that such interaction of income with shoe size would be theoretically meaningful, don't include shoe

size among the predictors. But, if you think there is some chance that a variable would be informative, include it.

Another cost of including numerous predictors is that "false alarms" may occur more often, such that predictors that really have no predictive value spuriously appear to have credibly non-zero regression coefficients. This problem can be addressed by using prior knowledge about relations among regression coefficients, as described in Section 17.2. By letting the predictors inform a hyperprior, they constrain each other's estimates, thereby producing shrinkage and attenuation of false alarms. If you have prior knowledge about the particular predictors, that your skeptical audience would assent to, then try to express it in the mathematical form of the prior.

Another cost of including numerous candidate predictors is that the noise and parameter flexibility introduced by the extra predictors can produce a loss of precision in the estimation of the coefficients. Even when the true regression coefficients on additional predictors are all zero, the uncertainty in their values can introduce uncertainty in the estimates of the original predictors. This bleeding of uncertainty is not always large, especially when the predictors are uncorrelated and there are many data points, but it does occur. You can experiment with this issue by using the random data generator in the program of Section 17.5.1 (MultipleLinearRegressionBrugs.R), and including different numbers of predictors with zero coefficients.

All of these issues are distinct from the peril of correlated predictors discussed at the beginning of the chapter. To reiterate, a predictor may appear to have a particular relation with the predicted variable, when other candidate predictors are left out of the analysis. But the apparent relation between the predictor and the outcome might be an illusory artifact, produced instead by the predictor being correlated with some other efficacious factor, while the putative predictor itself has zero or opposite influence on the outcome. Therefore it is important to include all reasonable predictors in the analysis, so that each has a better opportunity for being correctly interpreted, while also keeping in mind the costs mentioned in preceding paragraphs.

When considering the inclusion of interaction terms, and the goal of the analysis is explanation, then the main criterion is whether it is theoretically meaningful that the effect of one predictor should depend on the level of another predictor. Inclusion of an interaction term can also cause loss of precision in the estimates of lower-order terms. Moreover, interpretation of interactions and their lower-order terms can be subtle, as we saw, for example, in Figure 17.10. If the goal of the analysis is prediction, without emphasis on explanation, then interaction terms may be included to the extent that they enhance predictability without loss of parsimony. Bayesian model comparison can be useful in this case (see references cited at the end of this section).

Whenever an interaction term is included in a model, it is important to also include all lower-order terms. For example, if an x_i x_j interaction is included, then both x_i and x_j should also be included in the model. Although we did not discuss them, it is also possible to include three-way interactions such as x_i x_j x_k, if it is theoretically meaningful to do so. When this is done, it is important to include all the lower-order interactions and single predictors, including x_i x_j, x_i x_k, x_j x_k, x_i, x_j, and x_k. When the lower-order terms are omitted, this is artificially setting their regression coefficients to zero, and distorting the posterior estimates on the other terms. For clear discussion and examples of this issue, see Braumoeller (2004) and Brambor, Clark, and Golder (2006).

In some situations, it may be theoretically tenable to suppose that the candidate predictors have exactly zero influence on the variable to be predicted, and our goal is to iden-

tify which predictors have exactly zero influence. In this sense, we can establish a model-comparison framework in which different models have different regression coefficients fixed at zero. A large-scale Bayesian model comparison then tests which combinations of zero coefficients are most credible. There are numerous variations of this approach, and it is an active area of research (e.g., Casella & Moreno, 2006; Clyde & George, 2004; E. I. George, 2000; Greenland, 2008; Liang, Paulo, Molina, Clyde, & Berger, 2008; Scott & Berger, 2006).

17.5 R code

17.5.1 Multiple linear regression

This program was used to Figure 17.5, among others. Its definitions includes three different data sets. The rest of the program is designed to be generally applicable, so that the user can substitute other data sets without modifying the remainder of the program.

The use of the BUGS function `inprod()` allows the model specification to remain unchanged when the number of predictors changes. Unfortunately, `inprod()` is processed slowly by BUGS. If you adapt this program for a large application and it is running too slowly, try changing `inprod(b[],x[,])` to an explicit sum of products: `b[1]*x[i,1] + b[2]*x[i,2] + ...` .

(MultipleLinearRegressionBrugs.R)

```
graphics.off()
rm(list=ls(all=TRUE))
fname = "MultipleLinearRegressionBrugs"
library(BRugs)              # Kruschke, J. K. (2010). Doing Bayesian data analysis:
                              # A Tutorial with R and BUGS. Academic Press / Elsevier.
#------------------------------------------------------------------------------
# THE MODEL.

modelstring = "
# BUGS model specification begins here...
model {
    for( i in 1 : nData ) {
        y[i] ~ dnorm( mu[i] , tau )
        mu[i] <- b0 + inprod( b[] , x[i,] )
    }
    tau ~ dgamma(.01,.01)
    b0 ~ dnorm(0,1.0E-12)
    for ( j in 1:nPredictors ) {
        b[j] ~ dnorm(0,1.0E-12)
    }
}
# ... end BUGS model specification
" # close quote for modelstring
writeLines(modelstring,con="model.txt")
modelCheck( "model.txt" )

#------------------------------------------------------------------------------
# THE DATA.

dataSource = c("Guber1999","McIntyre1994","random")[1    ]

if ( dataSource=="Guber1999" ) {
```

```
    fname = "Guber1999" # file name for saved graphs
    dataMat = read.table( file="Guber1999data.txt" ,
                          col.names = c( "State","Spend","StuTchRat","Salary",
                                         "PrcntTake","SATV","SATM","SATT") )
    # Specify variables to be used in BUGS analysis:
    predictedName = "SATT"
    predictorNames = c( "Spend" , "PrcntTake" )
    #predictorNames = c( "Spend" , "PrcntTake" , "Salary" , "Stu       TchRat" )
    nData = NROW( dataMat )
    y = as.matrix( dataMat[,predictedName] )
    x = as.matrix( dataMat[,predictorNames] )
    nPredictors = NCOL( x )
}

if ( dataSource=="McIntyre1994" ) {
    fname = "McIntyre1994" # file name for saved graphs
    dataMat = read.csv(file="McIntyre1994data.csv")
    predictedName = "CO"
    predictorNames = c("Tar","Nic","Wt")
    nData = NROW( dataMat )
    y = as.matrix( dataMat[,predictedName] )
    x = as.matrix( dataMat[,predictorNames] )
    nPredictors = NCOL( x )
}

if ( dataSource=="random" ) {
    fname = "Random"  # file name for saved graphs
    # Generate random data.
    # True parameter values:
    betaTrue = c( 100 , 1 , 2 , rep(0,21) ) # beta0 is first component
    nPredictors = length( betaTrue ) - 1
    sdTrue = 2
    tauTrue = 1/sdTrue^2
    # Random X values:
    set.seed(47405)
    xM = 5 ; xSD = 2
    nData = 100
    x = matrix( rnorm( nPredictors*nData , xM , xSD ) , nrow=nData      )
    predictorNames = colnames(x) = paste("X",1:nPredictors,      sep="")
    # Random Y values generated from linear model with true param  eter values:
    y = x %*% matrix(betaTrue[-1],ncol=1) + betaTrue[1] + rnorm      (nData,0,sdTrue)
    predictedName = "Y"
    # Select which predictors to include
    includeOnly = 1:nPredictors # default is to include all
    #includeOnly = 1:10 # subset of predictors overwrites defau       lt
    x = x[,includeOnly]
    predictorNames = predictorNames[includeOnly]
    nPredictors = NCOL(x)
}

# Prepare data for BUGS:
# Re-center data at mean, to reduce autocorrelation in MCMC s    ampling.
# Standardize (divide by SD) to make prior specification eas       ier.
standardizeCols = function( dataMat ) {
      zDataMat = dataMat
      for ( colIdx in 1:NCOL( dataMat ) ) {
            mCol = mean( dataMat[,colIdx] )
            sdCol = sd( dataMat[,colIdx] )
            zDataMat[,colIdx] = ( dataMat[,colIdx] - mCol ) / sdCol
```

```
92        }
93        return( zDataMat )
94    }
95    zx = standardizeCols( x )
96    zy = standardizeCols( y )
97
98    # Get the data into BUGS:
99    datalist = list(
100                   x = zx ,
101                   y = as.vector( zy ) , # BUGS does not treat 1-column mat as vecto    r
102                   nPredictors = nPredictors ,
103                   nData = nData
104   )
105   modelData( bugsData( datalist ) )
106
107   #------------------------------------------------                    ----------------------------
108   # INTIALIZE THE CHAINS.
109
110   nChain = 3
111   modelCompile( numChains = nChain )
112
113   genInitList <- function(nPred=nPredictors) {
114       lmInfo = lm( datalist$y ~ datalist$x ) # R function returns ML          E
115       bInit = lmInfo$coef[-1]
116       tauInit = length(datalist$y) / sum(lmInfo$res^2)
117       list(
118           b0 = 0 ,
119           b = bInit ,
120           tau = tauInit
121       )
122   }
123   for ( chainIdx in 1 : nChain ) {
124       modelInits( bugsInits( genInitList ) )
125   }
126
127   #------------------------------------------------                    ----------------------------
128   # RUN THE CHAINS
129
130   # burn in
131   BurnInSteps = 100
132   modelUpdate( BurnInSteps )
133   # actual samples
134   samplesSet( c( "b0" , "b" , "tau" ) )
135   stepsPerChain = ceiling(10000/nChain)
136   thinStep = 2
137   modelUpdate( stepsPerChain , thin=thinStep )
138
139   #------------------------------------------------                    ----------------------------
140   # EXAMINE THE RESULTS
141
142   source("plotChains.R")
143   source("plotPost.R")
144
145   checkConvergence = F
146   if ( checkConvergence ) {
147     b0Sum   = plotChains( "b0"   , saveplots=F , filenameroot=fnam      e )
148     bSum    = plotChains( "b"    , saveplots=F , filenameroot=fname )
149     tauSum = plotChains( "tau" , saveplots=F , filenameroot=fn         ame )
150   }
```

```
151
152 # Extract chain values:
153 zb0Samp = matrix( samplesSample( "b0" ) )
154 zbSamp = NULL
155 for ( j in 1:nPredictors ) {
156     zbSamp = cbind( zbSamp , samplesSample( paste("b[",j,"]",      sep="") ) )
157 }
158 zTauSamp = matrix( samplesSample( "tau" ) )
159 zSigmaSamp = 1 / sqrt( zTauSamp ) # Convert precision to SD
160 chainLength = length(zTauSamp)
161
162 # Convert to original scale:
163 bSamp = zbSamp * matrix( sd(y)/apply(x,2,sd) , byrow=TRUE ,
164                          ncol=nPredictors , nrow=NROW(zbSamp) )
165 b0Samp = ( zb0Samp * sd(y)
166            + mean(y)
167            - rowSums( zbSamp
168            * matrix( sd(y)/apply(x,2,sd) , byrow=TRUE ,
169                      ncol=nPredictors , nrow=NROW(zbSamp) )
170            * matrix( apply(x,2,mean) , byrow=TRUE ,
171                      ncol=nPredictors , nrow=NROW(zbSamp) ) ) )
172 sigmaSamp = zSigmaSamp * sd(y)
173
174 # Save MCMC sample:
175 save( b0Samp , bSamp , sigmaSamp ,
176       file="MultipleLinearRegressionBrugsGuber1999.Rdata"       )
177
178 # Scatter plots of parameter values, pairwise:
179 if ( nPredictors <= 6 ) { # don't display if too many predictors
180     windows()
181     thinIdx = round(seq(1,length(zb0Samp),length=200))
182     pairs( cbind( zSigmaSamp[thinIdx] , zb0Samp[thinIdx] , zb       Samp[thinIdx,] )  ,
183       labels=c("Sigma zy","zIntercept",paste("zSlope",pred          ictorNames,sep="")))
184     windows()
185     thinIdx = seq(1,length(b0Samp),length=700)
186     pairs( cbind( sigmaSamp[thinIdx] , b0Samp[thinIdx] , bSam      p[thinIdx,] ) ,
187       labels=c( "Sigma y" , "Intercept", paste("Slope",predict          orNames,sep="")))
188     dev.copy2eps(file=paste(fname,"PostPairs.eps",sep="       "))
189 }
190 # Show correlation matrix on console:
191 cat("\nCorrlations of posterior sigma, b0, and bs:\n")
192 show( cor( cbind( sigmaSamp , b0Samp , bSamp ) ) )
193
194 # Display the posterior:
195 nPlotPerRow = 5
196 nPlotRow = ceiling((2+nPredictors)/nPlotPerRow)
197 nPlotCol = ceiling((2+nPredictors)/nPlotRow)
198 windows(3.5*nPlotCol,2.25*nPlotRow)
199 layout( matrix(1:(nPlotRow*nPlotCol),nrow=nPlotRow,n       col=nPlotCol,byrow=T) )
200 par( mar=c(4,3,2.5,0) , mgp=c(2,0.7,0) )
201 histInfo = plotPost( sigmaSamp , xlab="Sigma Value" , compV    al=NULL ,
202                      breaks=30 , main=bquote(sigma[y]) ,
203                      cex.main=1.67 , cex.lab=1.33 )
204 histInfo = plotPost( b0Samp , xlab="Intercept Value" , comp       Val=NULL ,
205                      breaks=30 , main=bquote(.(predictedName) *" at "* x==0) ,
206                      cex.main=1.67 , cex.lab=1.33 )
207 for ( sIdx in 1:nPredictors ) {
208 histInfo = plotPost( bSamp[,sIdx] , xlab="Slope Value" , co        mpVal=0.0 ,
209                      breaks=30 ,
```

```
                        main=bquote( Delta * .(predictedName) /
                                     Delta * .(predictorNames[sIdx]) ) ,
                        cex.main=1.67 , cex.lab=1.33 )
}
dev.copy2eps(file=paste(fname,"PostHist.eps",sep="")          ))

# Posterior prediction:
# Specify x values for which predicted y's are needed.
# xPostPred is a matrix such that ncol=nPredictors and nrow=      nPostPredPts.
xPostPred = rbind(
    apply(x,2,mean)-3*apply(x,2,sd) , # mean of data x minus th       rice SD of data x
    apply(x,2,mean)                 , # mean of data x
    apply(x,2,mean)+3*apply(x,2,sd)   # mean of data x plus thri       ce SD of data x
)
# Define matrix for recording posterior predicted y values f          or each xPostPred.
# One row per xPostPred value, with each row holding random pr   edicted y values.
postSampSize = chainLength
yPostPred = matrix( 0 , nrow=NROW(xPostPred) , ncol=postSa  mpSize )
# Define matrix for recording HDI limits of posterior predic          ted y values:
yHDIlim = matrix( 0 , nrow=NROW(xPostPred) , ncol=2 )
# Generate posterior predicted y values.
# This gets only one y value, at each x, for each step in the chai       n.
for ( chainIdx in 1:chainLength ) {
    yPostPred[,chainIdx] = rnorm( NROW(xPostPred) ,
                                  mean = b0Samp[chainIdx]
                                       + xPostPred %*% cbind(bSamp[chainIdx,]) ,
                                  sd = rep( sigmaSamp[chainIdx] , NROW(xPostPred) ) )
}
source("HDIofMCMC.R")
for ( xIdx in 1:NROW(xPostPred) ) {
    yHDIlim[xIdx,] = HDIofMCMC( yPostPred[xIdx,] )
}
cat( "\nPosterior predicted y for selected x:\n" )
show( cbind( xPostPred , yPostPredMean=rowMeans(yPostPr ed) , yHDIlim ) )

#------------------------------------------------          ----------------------------
```

17.5.2 Multiple linear regression with hyperprior on coe cients

This program was used to create Figure 17.7, among othe estheSe comments regarding inprod() before the previous program.

(MultiLinRegressHyperBrugs.R)

```
graphics.off()
rm(list=ls(all=TRUE))
fname = "MultiLinRegressHyper"
library(BRugs)              # Kruschke, J. K. (2010). Doing Bayesian dat       a analysis:
                              # A Tutorial with R and BUGS. Academic Press / Elsevier.
#------------------------------------------------          ----------------------------
# THE MODEL.

modelstring = "
# BUGS model specification begins here...
model {
    for( i in 1 : nData ) {
        y[i] ~ dnorm( mu[i] , tau )
        mu[i] <- b0 + inprod( b[] , x[i,] )
    }
```

```
      tau ~ dgamma(.01,.01)
      b0 ~ dnorm(0,1.0E-12)
      for ( j in 1:nPredictors ) {
          b[j] ~ dt( muB , tauB , tdfB )
      }
      muB ~ dnorm( 0 , .100 )
      tauB ~ dgamma(.01,.01)
      udfB ~ dunif(0,1)
      tdfB <- 1 + tdfBgain * ( -log( 1 - udfB ) )
}
# ... end BUGS model specification
" # close quote for modelstring
writeLines(modelstring,con="model.txt")
modelCheck( "model.txt" )

#------------------------------------------------- ----------------------------
# THE DATA.

tdfBgain = 1

dataSource = c("Guber1999","McIntyre1994","random")[3      ]

if ( dataSource=="Guber1999" ) {
    fname = paste("Guber1999Brugs","tdf",tdfBgain,sep="")
    dataMat = read.table( file="Guber1999data.txt" ,
                          col.names = c( "State","Spend","StuTchRat","Salary",
                                         "PrcntTake","SATV","SATM","SATT") )
    # Specify variables to be used in BUGS analysis:
    predictedName = "SATT"
    predictorNames = c( "Spend" , "PrcntTake" )
    #predictorNames = c( "Spend" , "PrcntTake" , "Salary" , "Stu        TchRat" )
    nData = NROW( dataMat )
    y = as.matrix( dataMat[,predictedName] )
    x = as.matrix( dataMat[,predictorNames] )
    nPredictors = NCOL( x )
}

if ( dataSource=="McIntyre1994Hyper" ) {
    fname = paste("McIntyre1994Brugs","tdf",tdfBgain,sep=       "")
    dataMat = read.csv(file="McIntyre1994data.csv")
    predictedName = "CO"
    predictorNames = c("Tar","Nic","Wt")
    nData = NROW( dataMat )
    y = as.matrix( dataMat[,predictedName] )
    x = as.matrix( dataMat[,predictorNames] )
    nPredictors = NCOL( x )
}

if ( dataSource=="random" ) {
    fname = paste("RandomHyper","tdf",tdfBgain,sep="")
    # Generate random data.
    # True parameter values:
    betaTrue = c( 100 , 1 , 2 , rep(0,21) ) # beta0 is first component
    nPredictors = length( betaTrue ) - 1
    sdTrue = 2
    tauTrue = 1/sdTrue^2
    # Random X values:
    set.seed(47405)
    xM = 5 ; xSD = 2
```

```
    nData = 100
    x = matrix( rnorm( nPredictors*nData , xM , xSD ) , nrow=nData )
    predictorNames = colnames(x) = paste("X",1:nPredictors, sep="")
    # Random Y values generated from linear model with true param eter values:
    y = x %*% matrix(betaTrue[-1],ncol=1) + betaTrue[1] + rnorm (nData,0,sdTrue)
    predictedName = "Y"
    # Select which predictors to include
    includeOnly = 1:nPredictors # default is to include all
    #includeOnly = 1:6 # subset of predictors overwrites defaul t
    x = x[,includeOnly]
    predictorNames = predictorNames[includeOnly]
    nPredictors = NCOL(x)
}

# Prepare data for BUGS:
# Re-center data at mean, to reduce autocorrelation in MCMC s ampling.
# Standardize (divide by SD) to make initialization easier.
standardizeCols = function( dataMat ) {
    zDataMat = dataMat
    for ( colIdx in 1:NCOL( dataMat ) ) {
        mCol = mean( dataMat[,colIdx] )
        sdCol = sd( dataMat[,colIdx] )
        zDataMat[,colIdx] = ( dataMat[,colIdx] - mCol ) / sdCol
    }
    return( zDataMat )
}
zx = standardizeCols( x )
zy = standardizeCols( y )

# Get the data into BUGS:
datalist = list(
        x = zx ,
        y = as.vector( zy ) , # BUGS does not treat 1-column mat as vecto r
        nPredictors = nPredictors ,
        nData = nData ,
        tdfBgain = tdfBgain
)
modelData( bugsData( datalist ) )

#------------------------------------------------ ----------------------------
# INTIALIZE THE CHAINS.

nChain = 3
modelCompile( numChains = nChain )

genInitList <- function(nPred=nPredictors) {
    lmInfo = lm( y ~ x ) # R function returns least-squares (normal MLE) fit.
    bInit = lmInfo$coef[-1] * apply(x,2,sd) / sd(y)
    tauInit = (length(y)*sd(y)^2)/sum(lmInfo$res^2)
    list(
        b0 = 0 ,
        b = bInit ,
        tau = tauInit ,
        muB = mean( bInit ) ,
        tauB = 1 / sd( bInit )^2 ,
        udfB = 0.95 # tdfB = 4
    )
}
for ( chainIdx in 1 : nChain ) {
```

```
134       modelInits( bugsInits( genInitList ) )
135   }
136
137   #------------------------------------------------                    ----------------------------
138   # RUN THE CHAINS
139
140   # burn in
141   BurnInSteps = 100
142   modelUpdate( BurnInSteps )
143   # actual samples
144   samplesSet( c( "b0" , "b" , "tau" , "muB" , "tauB" , "tdfB" ) )
145   stepsPerChain = ceiling(10000/nChain)
146   thinStep = 2
147   modelUpdate( stepsPerChain , thin=thinStep )
148
149   #------------------------------------------------                    ----------------------------
150   # EXAMINE THE RESULTS
151
152   source("plotChains.R")
153   source("plotPost.R")
154
155   checkConvergence = F
156   if ( checkConvergence ) {
157     b0Sum   = plotChains( "b0"   , saveplots=F , filenameroot=fnam      e )
158     bSum    = plotChains( "b"    , saveplots=F , filenameroot=fname )
159     tauSum = plotChains( "tau" , saveplots=F , filenameroot=fn        ame )
160     muBSum = plotChains( "muB" , saveplots=F , filenameroot=fn    ame )
161     tauBSum = plotChains( "tauB" , saveplots=F , filenameroot=       fname )
162     tdfBSum = plotChains( "tdfB" , saveplots=F , filenameroot=       fname )
163   }
164
165   # Extract chain values:
166   zb0Samp = matrix( samplesSample( "b0" ) )
167   zbSamp = NULL
168   for ( j in 1:nPredictors ) {
169       zbSamp = cbind( zbSamp , samplesSample( paste("b[",j,"]",      sep="") ) )
170   }
171   zTauSamp = matrix( samplesSample( "tau" ) )
172   zSigmaSamp = 1 / sqrt( zTauSamp ) # Convert precision to SD
173   chainLength = length(zTauSamp)
174
175   # Convert to original scale:
176   bSamp = zbSamp * matrix( sd(y)/apply(x,2,sd) , byrow=TRUE ,
177                            ncol=nPredictors , nrow=NROW(zbSamp) )
178   b0Samp = ( zb0Samp * sd(y)
179              + mean(y)
180              - rowSums( zbSamp
181              * matrix( sd(y)/apply(x,2,sd) , byrow=TRUE ,
182                        ncol=nPredictors , nrow=NROW(zbSamp) )
183              * matrix( apply(x,2,mean) , byrow=TRUE ,
184                        ncol=nPredictors , nrow=NROW(zbSamp) ) ) )
185   sigmaSamp = zSigmaSamp * sd(y)
186
187   # Scatter plots of parameter values, pairwise:
188   if ( nPredictors <= 6 ) { # don't display if too many predictors
189       windows()
190       thinIdx = round(seq(1,length(zb0Samp),length=200))
191       pairs( cbind( zSigmaSamp[thinIdx] , zb0Samp[thinIdx] , zb        Samp[thinIdx,] )   ,
192         labels=c("Sigma  zy","zIntercept",paste("zSlope",pred          ictorNames,sep="")))
```

```
193       windows()
194       thinIdx = seq(1,length(b0Samp),length=700)
195       pairs( cbind( sigmaSamp[thinIdx] , b0Samp[thinIdx] , bSam       p[thinIdx,] ) ,
196         labels=c( "Sigma y" , "Intercept", paste("Slope",predict          orNames,sep="")))
197       dev.copy2eps(file=paste(fname,"PostPairs.eps",sep="        "))
198   }
199   # Show correlation matrix on console:
200   cat("\nCorrlations of posterior sigma, b0, and bs:\n")
201   show( cor( cbind( sigmaSamp , b0Samp , bSamp ) ) )
202
203   # Display the posterior:
204   nPlotPerRow = 5
205   nPlotRow = ceiling((2+nPredictors)/nPlotPerRow)
206   nPlotCol = ceiling((2+nPredictors)/nPlotRow)
207   windows(3.5*nPlotCol,2.25*nPlotRow)
208   layout( matrix(1:(nPlotRow*nPlotCol),nrow=nPlotRow,n       col=nPlotCol,byrow=T) )
209   par( mar=c(4,3,2.5,0) , mgp=c(2,0.7,0) )
210   histInfo = plotPost( sigmaSamp , xlab="Sigma Value" , compV    al=NULL ,
211                        breaks=30 , main=bquote(sigma[y]) ,
212                        cex.main=1.67 , cex.lab=1.33 )
213   histInfo = plotPost( b0Samp , xlab="Intercept Value" , comp       Val=NULL ,
214                        breaks=30 , main=bquote(.(predictedName) *" at "* x==0) ,
215                        cex.main=1.67 , cex.lab=1.33 )
216   for ( sIdx in 1:nPredictors ) {
217   histInfo = plotPost( bSamp[,sIdx] , xlab="Slope Value" , co         mpVal=0.0 ,
218                        breaks=30 ,
219                        main=bquote( Delta * .(predictedName) /
220                                     Delta * .(predictorNames[sIdx]) ) ,
221                        cex.main=1.67 , cex.lab=1.33 )
222   }
223   dev.copy2eps(file=paste(fname,"PostHist.eps",sep=""        ))
224
225   # Posterior prediction:
226   # Specify x values for which predicted y's are needed.
227   # xPostPred is a matrix such that ncol=nPredictors and nrow=     nPostPredPts.
228   xPostPred = rbind(
229       apply(x,2,mean)-3*apply(x,2,sd) , # mean of data x minus th       rice SD of data x
230       apply(x,2,mean)                 , # mean of data x
231       apply(x,2,mean)+3*apply(x,2,sd)   # mean of data x plus thri       ce SD of data x
232   )
233   # Define matrix for recording posterior predicted y values f          or each xPostPred.
234   # One row per xPostPred value, with each row holding random pr  edicted y values.
235   postSampSize = chainLength
236   yPostPred = matrix( 0 , nrow=NROW(xPostPred) , ncol=postSa  mpSize )
237   # Define matrix for recording HDI limits of posterior predic          ted y values:
238   yHDIlim = matrix( 0 , nrow=NROW(xPostPred) , ncol=2 )
239   # Generate posterior predicted y values.
240   # This gets only one y value, at each x, for each step in the chai       n.
241   for ( chainIdx in 1:chainLength ) {
242       yPostPred[,chainIdx] = rnorm( NROW(xPostPred) ,
243                                     mean = b0Samp[chainIdx]
244                                            + xPostPred %*% cbind(bSamp[chainIdx,]) ,
245                                     sd = rep( sigmaSamp[chainIdx] , NROW(xPostPred) ) )
246   }
247   source("HDIofMCMC.R")
248   for ( xIdx in 1:NROW(xPostPred) ) {
249       yHDIlim[xIdx,] = HDIofMCMC( yPostPred[xIdx,] )
250   }
251   cat( "\nPosterior predicted y for selected x:\n" )
```

```
252 show( cbind( xPostPred , yPostPredMean=rowMeans(yPostPred) , yHDIlim ) )
253
254 #------------------------------------------------- -----------------------------
```

17.6 Exercises

Exercise 17.1. [Purpose: View the prior on the regression coefficients, when there is a hyperprior.] The hyperprior on regression coefficients in Figure 17.6 may be difficult to intuit. Therefore, generate graphs of prior distribution on the regression coefficients for the program in Section 17.5.2 (MultiLinRegressHyperBrugs.R), when tdf is set at different values. To accomplish this, do the following:

1. Because BUGS balks at extreme values of tau, and extreme values can be sampled under the vague prior, the gamma distributions for tau need to be censored. Therefore, change the tau specifications to

```
tau  dgamma(.01,.01)I(0.0001,10000)
tauB  dgamma(.01,.01)I(0.0001,10000)
```

2. In the datalist, comment out only the single line that specifies the y values. The other lines, that specify the x values and the number of predictors, etc., must remain, because they specify the structure of the model. The model only predicts y as a function of x; the model does not predict x.

3. In the initialization of the chains, comment out the data-based initialization, and instead use modelGenInits().

4. Do not use the plotChains() command, because some of the MCMC values may be too extreme for some of the BUGS graphics routines that are called by plotChains(), and BUGS will crash.

5. When plotting the slope values in the last part of the code, manually specify the axis limits so that you can see the central parts of the distribution, perhaps like this:

```
histInfo = plotPost( bSamp[,sIdx] , xlab="Slope Value" , compVal=0.0 ,
        breaks=c(-1000000,seq(-400,400,21),1000000) , xlim=c(-400,400) , ...)
```

Run the program and display the prior for tdfBg=1 and for tdfBg=100. Point out and discuss any differences in the priors for those different hyperprior constants.

Exercise 17.2. [Purpose: Power analysis for multiple regression.] Consider the SAT data shown in Figure 17.3, p. 374, and the posterior for a linear regression model (with no interaction), shown in Figure 17.5, p. 377. The marginal posterior distribution for the slope on spending-per-pupil has a 95% HDI that excludes zero, and that has a width very nearly 17. Consider two different goals for the research: One goal is to show that the 95% HDI on spending-per-pupil excludes a ROPE of [−1, +1], and the second goal is to show that the width of the 95% HDI is less than 10. How can we assess the probability of achieving these goals?

To address the question, we think of the y values as being randomly generated according to the linear regression model, at the x values that are provided by the world. The model only describes the dependency of y on x; the model does not describe the distribution of the x values. In different applications the x values have different actual generators. In experiments, the x values are selected by the experimenter, and power analysis explicitly manipulates the values of the xs and the frequencies of each. In observational studies, the x values are generated by the world, not by the experimenter. In some observational studies, we can think of the x values as randomly drawn from some underlying distribution (e.g., x could be a person's height, and we can easily randomly sample another person to get

another x value. In other observational studies, it is a conceptual stretch to think of x as being a random value that is sampled from some underlying population; e.g., spending-per-pupil in each of the 50 states. But even in that last case, we think of each state as being representative of some universe of possible states, from which 50 were drawn.

In the present non-experimental, observational study, (i.e. the SAT data for 50 states), the x values are not selected by researcher. Moreover, we have no model of the x values. Therefore, we use the actual x values themselves as the best available description of the underlying distribution of x values. When we simulate a new datum, we first randomly select one of the points from the actual x values, and then we randomly generate a y value according the credible parameter values in the model.

In our specific situation there are only 50 states altogether, and therefore we can do a retrospective power analysis using a sample size of 50. But we can also imagine increasing the sample size if, instead of using state-average data, we imagine that the data are from individual school districts within states. Under this interpretation, the posterior from the state-average data, and the range of x values, might not be fully representative of district data, but at least it's better than nothing.

Answer these questions: What is the retrospective power for each of the two goals? (Hint: It's about .67 and .00, when the x points are randomly sampled with replacement from the 50 actual points.) What is the power for each of the two goals when $N = 130$? (Hint: It's about .85 and .30.) What's the minimal N needed to achieve a power of .80 for the second goal?

Programming hints: Use the power-analysis program of Section 13.6.2 (FilconBrugsPower.R) as a template for adapting the linear regression program of Section 17.5.1 (MultipleLinearRegressionBrugs.R). You'll want to pass the raw data into the GoalAchievedForSample() function because that way the data can be standardized for JAGS but the resulting parameter values can be converted back to original scale (using information from the original data).

Chapter 18

Metric Predicted Variable with One Nominal Predictor

Contents

> Familywise error rates breed rumors of incest,
> Hounding for quarry in multiple t tests.
> Barking at research, poor dog got run over;
> Should have done Bayesian oneway ANOVA.

In this chapter we consider situations with a metric predicted variable and a nominally-scaled predictor variable. There are many cases of these situations in real-world research. For example, we may want to predict weight loss (a metric variable) as a function of which diet the person follows (e.g., low-carb, vegetarian, or low-fat). As another example, we may want to predict severity of psychosis (measured on a metric scale) as a function of which anti-psychotic drug the person takes. Or we may want to predict income as a function of political party affiliation. This combination of predicted and predictor scales occurs in the first row, fourth cell, of Table 14.1 (p. 312).

In traditional NHST analyses, these situations are addressed by "oneway analysis of variance" (ANOVA). The term "oneway" refers to the fact that a single nominal variable is being used as the predictor. The phrase "analysis of variance" refers to the fact that the overall variance among all the values is decomposed, analyzed, into two parts: variance within the levels of the nominal predictors, and variance between the levels of the nominal

predictors. The variance within levels of the nominal predictor is called noise or error, i.e., variability that cannot be predicted by the predictor. The complementary variance between the levels of the nominal predictor is called the effect of the predictor. Usually we do the research with the goal of detecting an effect, which means that we would like the magnitude of the variance between levels to be large compared to the variance within levels. The ratio, of variance between to variance within, is called the F ratio. In the Bayesian approach, we rarely if ever refer to the F-ratio. But because the model we use is based on the model of traditional ANOVA, we will refer to our analysis as Bayesian ANOVA, or sometimes BANOVA.

18.1 Bayesian oneway ANOVA

The basic idea of oneway ANOVA was introduced in Section 14.1, p. 300. The predictor is a variable measured on a nominal scale. For example, one might predict y as a function of political party affiliation, notice that the predictor has nominal levels such as libertarian, green, democratic, republican, and so on. We denote the predictor variable as $\mathbf{x}$, which is a vector with one component per nominal level. For example, suppose that the predictor is political party affiliation, with Green as level 1, Democrat as level 2, Republican as level 3, Libertarian as level 4, and Other as level 5. Then Democrat is represented as $\mathbf{x} = \langle 0,1,0,0,0 \rangle$, and Libertarian is represented as $\mathbf{x} = \langle 0,0,0,1,0 \rangle$. Political party affiliation is being treated here as a categorical label only, with no ordering along a liberal-conservative scale.

The formal model indicates how to derive the predicted value from the predictor. The idea is that there is a baseline quantity of the predicted variable, and each level of the predictor indicates a deflection above or below that baseline. We will denote the baseline value of the prediction as β_0. The deflection for the j^{th} level of the predictor is denoted β_j. When the predictor has value $\mathbf{x}_i = \langle \ldots, x_{ji}, \ldots \rangle$, then the predicted value is

$$\begin{aligned} \mu_i &= \beta_0 + \sum_j \beta_j x_{ji} \\ &= \beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i \end{aligned} \tag{18.1}$$

where the notation $\boldsymbol{\beta} \cdot \mathbf{x}$ denotes the "dot product" of the vectors. In Equation 18.1, the coefficient β_j indicates how much y changes when $\mathbf{x}$ changes from neutral to level j. In other words, β_j indicates how much y changes when $\mathbf{x}$ changes from all $x_j = 0$ to $x_j = 1$. The baseline is constrained such that the deflections sum to zero across the levels of $\mathbf{x}$:

$$\sum_{j=1} \beta_j = 0 \tag{18.2}$$

The expression of the model in Equation 18.1 is not complete without the constraint in 18.2. Examples were shown in Figure 14.4, p. 301, and it is worth the effort to go now to that Figure for a quick review.

The predicted value, μ_i, in Equation 18.1 is of the central tendency in the data. The data themselves are assumed to be randomly generated around that central tendency. As usual, we will assume a normal distribution, $y_i \sim N(\mu_i, \tau)$, where τ is the precision of the normal distribution. As discussed in previous chapters, if the data have outliers, a t distribution may be used instead.

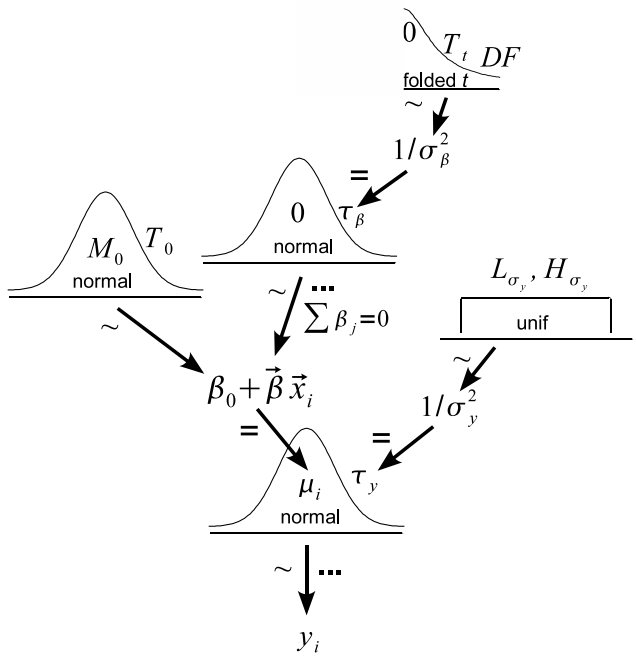

Figure 18.1: Hierarchical dependencies for model of oneway Bayesian ANOVA. The baseline is β_0, and the deflection away from that baseline for the j^{th} level of x is β_j. The standard deviation of the β_j's has a folded-t prior. The variance within levels of x is estimated by the precision τ_y, which is here assumed to be homogeneous across groups, although it need not be in general.

18.1.1 The hierarchical prior

Our primary interest is in estimating the deflections, β_j, for each level of x. We could just put a separate prior on each parameter and estimate them separately from each other. It is typical, however, that the levels of x are not utterly unrelated to each other, and therefore data from one level may inform estimates in another level. For example, the deflections for republicans, libertarians, and greens can inform an estimate of the deflection for democrats. Thus, if the deflection for libertarians is -1.0, for republicans is $+0.5$, and for greens is 1.0, then the deflection for democrats should be somewhere in the general range, and not out at, say, 12.0. At the least, we might have prior beliefs that the deflections for most levels of x may be small, with only a few deflections being large, and therefore we can let the various levels mutually inform each other's estimates based on this structural assumption.

The form of the hierarchical model for oneway BANOVA is displayed in Figure 18.1 (Gelman, 2005, 2006). In the upper middle of the diagram is a normal distribution that describes the distribution of deflections, β_j, across levels of x. This normal distribution has a mean at zero, reflecting the fact that the deflections are constrained to fall both above and below the baseline, because they must sum to zero. And, the precision of this normal distribution, τ_β, is estimated, not pre-set at a constant. Thus, if many of the levels of x have a small deflection in the data, then the precision is estimated to be high, and this in turn shrinks the estimates of other β_j.

The prior for τ_β derives from the recommendation of Gelman (2006). First, the preci-

sion is converted to standard deviation: = 1= ^{2}. Then a folded-t distribution is used as a prior on . The folded-t is just the positive side of the usual-t distribution. Notice that it is defined only over non-zero values, as is required for, and it extends to positive infinity. Unlike the gamma( ) distribution that is often used for precisions, however, the folded-t does not have infinite density near zero. Because noisy data never rule out detections of all zero, there can cause unintended distortions in the estimates if the prior places extreme densities at either end of the scale (see Gelman, 2006, for details).

A folded-t prior could also be used for the noise, , but we will use a uniform, again as recommended by Gelman (2006). One motivation is that a uniform may have a more intuitive form than a folded-t when expressing a prior belief. A second reason is that the infelicities of estimation that affect are not present so prominently at this level in the model, because the within-level noise is typically not near zero, and there are enough data points to overwhelm any mildly informed prior.

18.1.1.1 Homogeneity of variance

The model described here assumes equal variances across all levels of . As a concrete example, the model assumes that the variance of income is the same for republicans as for democrats as for libertarians as for greens. This assumption of homogeneous variances is vestigial from two precursors. First, the analogous assumption is made in linear regression, and ANOVA may be construed, mathematically, as a special case of linear regression. Second, homogeneity of variance is assumed in NHST ANOVA to simplify derivation of F distributions. Neither of these precursors actually demands that we make the assumption of equal variances in BANOVA.

The model described here assumes homogeneity of variance only for simplicity in presentation. By assuming equal variances for all levels of , the focus could be on the estimation of the group deviations parameters, . Also, by assuming equal variances, the results of BANOVA can be more directly compared to the results of NHST ANOVA, if such a comparison is desired.

In principle, the BANOVA model can (and should) estimate different variances for each level of . The model in Figure 18.1 can be expanded analogously to the model in Figure 16.11, p. 356. Instead of a single precision, , used for all levels of , a separate precision is estimated for each level of , as in the lower-right of Figure 16.11. A higher-level distribution describes the spread of the across levels of . This structure provides shrinkage of the estimates of the , to the extent that the data suggest homogeneity of variance. Exercise 18.3 has you give this scheme a test drive.

18.1.2 Doing it with R and BUGS

As usual, every arrow in the hierarchical diagram of Figure 18.1 has a corresponding line in the BUGS model specification. The parameters that appear as in Figure 18.1 are denoted by "a[j]" in the model specification.

To understand the way that the model is specified in the BUGS script, it is important to understand how the data are formatted. The values in the program are coded as integer indices 1, 2, 3, ..., and not as vectors (1; 0; 0; : : :), (0; 1; 0; : : :), (0; 0; 1; : : :), ... By coding as integers, then nested indexing can be used instead of products of vectors. Thus, becomes coded as a[x] , not inprod(a[],x[]) . For the i^{th} observation, the value of

is coded as x[i] . Thus, x[i] 2 f1; 2; 3; : : : ; NxLvlg for i 2 f1; : : : ; Ntotal g, where NxLvl is the number of levels of x and Ntotal is the total number of observations.

Here is the BUGS model speci cation: (ANOVAonewayBRugs.R)

```
model {
  for ( i in 1:Ntotal ) {
    y[i] ~ dnorm( mu[i] , tau )
    mu[i] <- a0 + a[x[i]]
  }
  #
  tau <- pow( sigma , -2 )
  sigma ~ dunif(0,10) # y values are assumed to be standardized
  #
  a0 ~ dnorm(0,0.001) # y values are assumed to be standardized
  #
  for ( j in 1:NxLvl ) { a[j] ~ dnorm( 0.0 , atau ) }
  atau <- 1 / pow( aSD , 2 )
  aSD <- abs( aSDunabs ) + .1
  aSDunabs ~ dt( 0 , 0.001 , 2 )
}
```

The constraint, that the de ections sum to zero, does not appear in the model speci cation. The BUGS code estimates the baseline and de ections without the constraint, but the MCMC estimates are re-centered at zero by subsequent R code. The non-centered baseline is denoted in the BUGS model as a0, and the non-centered de ections are denoted a[j]. Those non-centered estimates are transformed to respect the sum-to-zero constraint merely by subtracting the mean of the a[j] 's from each a[j] , and adding the mean to the baseline. Thus, b[j] = a[j] - mean(a) and b0 = a0 + mean(a)

The constants for the top-level priors are set with the assumption that the data values, y, have been standardized. according to Equation 16.1, p. (Of course, the x values cannot be standardized because they are nominal.) This standardization makes it easier to establish reasonable default priors for a range of applications, without having to change the prior constants when the application changes, for example from income, on the order of 10^5 dollars, to width of hairs, on the order of 10^{-1} millimeters. Nevertheless, when there is strong prior information, it should be incorporated. Exercise 18.2 has you explore robustness of the results when you use different priors.

There is one other trick in the BUGS model speci cation that is not in the hierarchical diagram of Figure 18.1. One line of the BUGS model speci es that the standard deviation of the group e ects, denoted aSDunabs, comes from a t distribution: aSDunabs dt(0,0.001,2) . Another line takes the absolute value to "fold" the distribution onto the non-negative numbers, yielding the value aSD aSD <- abs(aSDunabs) + .1. But that line also mysteriously adds a small constant, namely 0.1. This constant keeps aSD from venturing extremely close to zero. The reason for keeping aSD away from zero is that shrinkage can become overwhelmingly strong when there are many groups with few data per group. This becomes especially problematic in the next chapter when we consider interaction of factors.

It turns out that MCMC sampling for this model can be extremely inef cient. One important way to reduce burn-in time is to start the chains at reasonable positions. We start the overall baseline at the grand mean of the data, and the de ections at the level means minus the grand mean. The variances are also initialized near the corresponding data variances. The full code, including initialization of chains, is presented in Section 18.4.1 (ANOVAonewayBRugs.R).

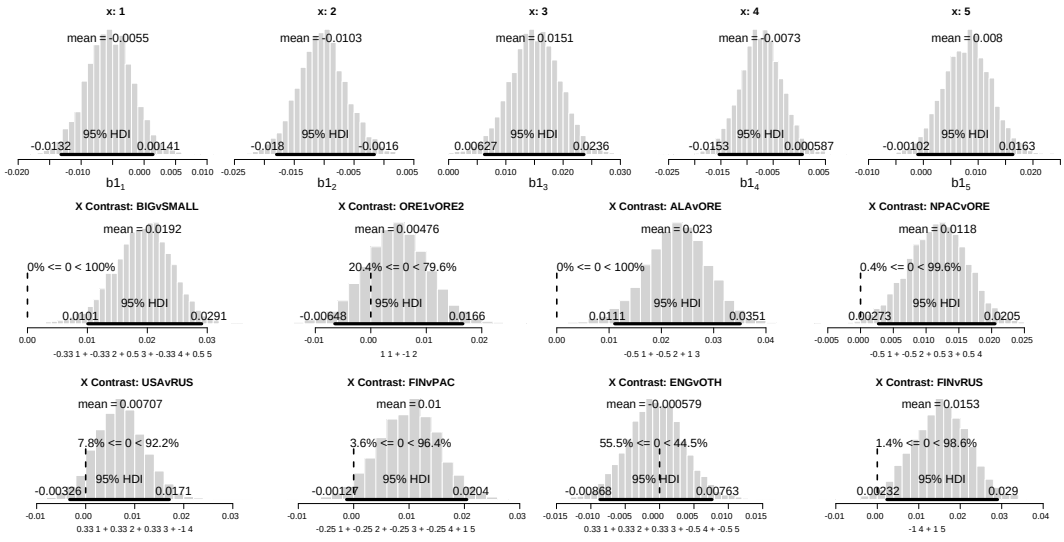

Figure 18.2: Upper row: Posterior estimates of β_j values for data from McDonald (2009; McDonald et al., 1991). The values indicate deflections by each group away from the overall central tendency. Middle and bottom rows: Various complex comparisons of β_j values. For example, the bottom row, third panel, compares the three English-speaking sites against the two non-English speaking sites (where they might say "midiya myshtsy" or "simpukka lihas" instead of "mussel muscle").

Because the chains can be highly autocorrelated, extensive thinning is needed, keeping a step only once out of several hundred. Running such long chains can take a long time, and become boring for your computer, which would rather be searching the web for exciting software updates. In the examples presented here, we tolerate the modest waiting times. But there are various methods for re-parameterizing the models so that the chains are sampled with less auto-correlation (e.g., Gelman, 2006; Gelman & Hill, 2007, Ch. 19).

18.1.3 A worked example

With all the emphasis these days on physical fitness and muscle building, it's only appropriate to consider an example about muscles. In particular, I'd like to know if geographical location influences muscle size, which might be affected by the weather or amount of daylight. Consider some data regarding muscles from five geographical locations: (1) Tillamook, Oregon; (2) Newport, Oregon; (3) Petersburg, Alaska; (4) Magadan, Russia (Pacific coast); and (5) Tvarminne, Finland. The values in the data set are the length of the anterior adductor muscle scar divided by total muscle length, in the species Mytilus trossulus. These ratios of scar length to total length tend to be between 5% and 15% (McDonald, 2009; McDonald, Seed, & Koehn, 1991).

Results of the BRugs program listed in Section 18.4.1 (OneWayAonewayBRugs.) are shown in Figure 18.2. The histograms in the upper row show the (marginal) posterior distributions of the β_j values for the five geographical locations. These β_j values are deflections away from the baseline β_0, which is not shown. Some things to keep in mind when interpreting the results: First, the estimates of deflection are subject to "shrinkage", because the model incorporates the prior structural assumption that all the deflections come from the same

overarching distribution. The mean deviations shown in Fig. 18.2 are, in fact, a little smaller than the deviations of the actual sample means. Second, the model assumes that the precision is the same for all groups; i.e., there is homogeneity of variance. The posterior μ_j values are the ones that are believable when also assuming homogeneous variances. If the groups actually have wildly different variances, then the estimates for μ_j may be distorted. Third, the marginal distributions on the μ_j cannot be used to directly infer differences between groups, because the parameters might be correlated. Indeed, the deviations tend to be negatively correlated, because increasing the estimated deviation for one group suggests decreasing the estimated deviation for another, if they are to remain symmetric around the baseline. To judge differences between groups, the differences must be computed directly.

18.1.3.1 Contrasts and complex comparisons

The middle and bottom rows of Figure 18.2 shows several comparisons for the mussel muscle results. A comparison amounts to a difference between an average of some groups and an average of other groups. For example, to compare the Pacific Ocean mussels against the one non-Pacific (Baltic Sea) mussel, we multiply the deviations (μ_j's) of the first four groups by 1/4 to get their average, and subtract it from the deviation (μ_5) of the fifth group to get the difference. The difference is called a *contrast*, and when the comparison involves a contrast of averages, instead of a contrast of two specific groups, it is sometimes called a *complex* comparison. The contrast is fully specified by the coefficients on the groups, which can be placed into a vector of *contrast coefficients*. For example, the contrast coefficients for comparing Pacific Ocean mussels against Baltic Sea mussels are $1/4; 1/4; 1/4; 1/4; -1$. Notice that the coefficients sum to zero. We compute the difference at every step in the MCMC chain, and examine the resulting distribution of believable differences. The distribution for this particular example is shown in the bottom row, second panel, of Figure 18.2, where it can be seen that over 96% of the believable differences lie on one side of zero, and the 95% HDI just spans zero. From these results we may not want to declare categorically that there is a credible difference between Finland and the other sites; the decision depends on how you set your ROPE. Regardless of your decision rule, the posterior does tells us the most credible difference and the uncertainty in that difference.

Figure 18.2 shows a variety of comparisons that might be of interest. For example, the first panel of the middle row compares the two sites with the largest muscles against the three other sites. This sort of comparison would be labeled "post hoc" by traditional analyses, because we might not have specified which sites would be suggestive before collecting the data. The second panel in the middle row contrasts the two sites in Oregon. The third panel in the middle row compares the Alaska site against the average of the two Oregon sites.

We can make all the comparisons shown in Figure 18.2, and as many others as we like, without worrying about inflated false alarm rates, because the posterior distribution does not change when we consider additional comparisons. The posterior distribution is the best inference we can make based on the data we have and the beliefs we started with. It is possible that the random data in our sample are wildly unrepresentative of the underlying population, but we cannot know. Fortunately, because of the incorporation of our prior knowledge about how estimates in the different locations can mutually inform each other, the estimates undergo shrinkage, which helps mitigate the effect of rogue data. In many applications, the shrinkage yields decisions similar to those that would result from NHST "corrections" for multiple comparisons. But unlike NHST corrections, the

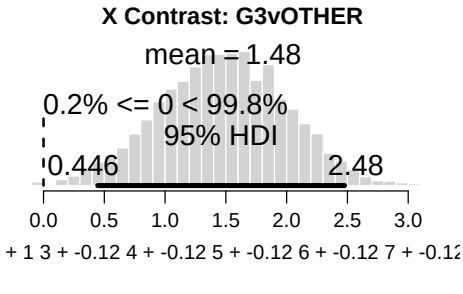

Figure 18.3: A comparison of Group 3 versus the average of other groups, for the data in Solari et al. (2008, Table 3, p. 495). (The specification of contrast coefficients on the x-axis overflows the margins of the figure because there are so many groups. The contrast coefficients on the nine groups are $1=8; 1=8; +1; 1=8; :::$, which, when rounded to two decimal places, appear as $0:12; 0:12; +1; 0:12; :::$.)

shrinkage in the Bayesian approach is based on explicit prior knowledge, and is not affected by which or how many comparisons are intended. (For more discussion of these issues, see Section 17.2, regarding decisions about multiple regression coefficients, and Section 11.4, regarding multiple comparisons of groups.)

18.1.3.2 Is there a difference?

The contrasts and complex comparisons in Figure 18.2 were judged to be credibly non-zero if the 95% HDI excluded (a ROPE around) zero. A difference would be deemed to be practically equivalent to zero if its HDI fell entirely within a ROPE. This decision procedure is attractive because all the groups are simultaneously estimated, with mutually informed shrinkage, and from priors that are also appropriately informed (which entails also being agreeable to a skeptical audience).

Some researchers prefer to pose the question, "Is there a difference?", as a model comparison on two priors. One prior expresses the null hypothesis that the contrast has zero magnitude, while the other prior expresses a complementary hypothesis that any magnitude contrast is possible. This approach was discussed extensively in Section 12.2, beginning on p. 245.

There are two attractions to the two-prior, model-comparison approach. One attraction is that the model comparison can yield posterior odds in favor of the null, unlike NHST, which can only reject a null hypothesis but never accept it. Another attraction is that the complementary hypothesis is usually intended to be an "automatic" uninformed prior that is chosen for mathematical felicity. The hope is that an automatic prior obviates debate about how prior information should be expressed.

As was argued in Section 12.2, the two-prior approach should be applied cautiously. First, it is important to emphasize that the two-prior approach only indicates which prior is relatively less unbelievable. If either prior is theoretically untenable in the first place, then the "automatic" model comparison is automatically uninformative. Thus, the two-prior approach should only be applied to situations in which (i) it is theoretically appropriate to posit that a particular contrast really can be exactly zero, and (ii) the alternative prior incorporates prior knowledge about the plausible magnitudes of the difference.

As an example, consider a situation presented by Solari, Li, and Sun (2008, Table 3, p. 495). There were nine groups, with a metric dependent variable. The dependent variable was the acetic acid content of tomatoes, and the nine groups were different types of manuring during growth of the tomatoes. The mean of Group 3 appeared to be different than other groups. To test whether Group 3 was different, the authors conducted a Bayesian model comparison of two priors: The null-hypothesis prior had all nine groups with iden-

tical means. The alternative prior had Group 3 with a separate estimated mean while the other eight groups had identical means. The resulting Bayes factor (BF) strongly favored the alternative prior. Does this result suggest that the alternative prior is what we should believe? Unfortunately, no. The BF tells us that the prior with eight equal means and one different mean for Group 3 is more believable than the prior with nine equal means (assuming that the priors on the two hypotheses were 50-50). But the prior with eight equal means, on groups other than Group 3, is already untenable because we do not believe that the eight groups have identical means. Moreover, the estimate of difference, between Group 3 and the other groups, is not what we want, because the estimate does not take into account variation among the eight other groups.

When instead we conduct a Bayesian analysis using the BANOVA model, we obtain a posterior that simultaneously estimates all the separate group means, with shrinkage, from a plausibly informed prior. The complex comparison of Group 3 against the other eight groups is shown in Figure 18.3, where it can be seen that the magnitude of the contrast is credibly greater than zero. In this application, there is no need to pursue a BF approach to group comparisons.

It is also worth reiterating that the two-prior, model-comparison approach can arrive at a conclusion opposite that of the one-prior, estimation approach. Recall Figure 12.5, p. 249, which showed that a model comparison preferred the hypothesis of identical groups to the alternative hypothesis of all different groups, even though an estimation of effects in the alternative hypothesis showed a credible difference among groups. The point in that case was that the null model, even though it was a poor model, was less bad than the alternative model. Follow-up model comparisons would be required to narrow down which combination of group equivalences was least implausible. And even after that, we would not necessarily want to believe that any of the groups are truly equivalent, because we know in advance that they were treated differently. Instead, we desire an estimate of the differences and the precision of the estimate. That situation involved a dichotomous dependent variable, but the analogous situation can arise for metric dependent variables.

The two-prior, model-comparison approach can be appropriate in situations where actual equivalence is tenable and the goal is to identify which conditions are plausibly equivalent, or situations in which zero-magnitude effects are tenable and the goal is to identify which conditions have zero effect. In those situations, it behooves the researcher to pursue the model-comparison or related approaches (see, e.g., Berger & Sellke, 1987; Hochberg, 1999; Gopalan & Berry, 1998; Mueller et al., 2007; Scott & Berger, 2006). Moreover, Bayesian model comparison is highly advisable when the two models are genuinely viable competitors that express different explanations of the data. In these situations, it is important that the priors in the two models are equivalently informed, so that neither model is at a disadvantage because of an infelicity in an arbitrary, automatic prior.

18.2 Multiple comparisons

In twentieth century null-hypothesis significance testing (NHST), there is an immense literature regarding how to compute the "true" significance (i.e., probability of false alarm) of an apparent difference between groups, when the analyst is conducting comparisons of multiple groups. The problem is that when more comparisons are conducted, there are more opportunities for a spuriously large difference to appear by accident. In other words, there are more opportunities for false alarms. Notice that this problem of inflated false alarm

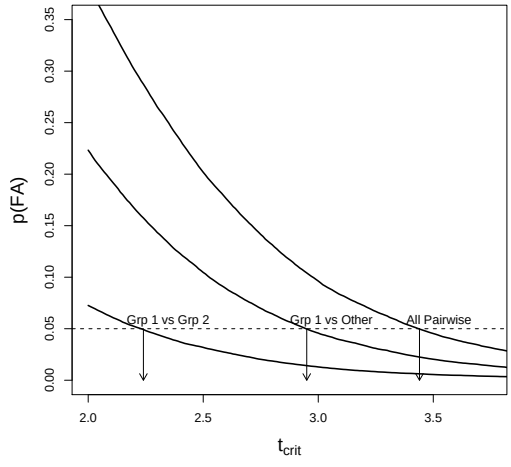

Figure 18.4: The probability of false alarm as a function of critical t value, with separate curves for different sets of comparisons. All groups have $N = 6$ fixed by intention. The curve labeled "Grp 1 vs Grp 2" is for a single comparison of two groups, and corresponds with the usual two-group t distribution. The curve labeled "Grp 1 vs Other" refers to four paired comparisons, of Group 1 versus each of the other four groups. The last curve is for the set of all 10 paired comparisons.

rates arises because NHST is based on the intentions of the analyst. If the analyst intends to make lots of comparisons between various combinations of groups, then there is greater opportunity for false alarms. If the analyst intends to make only a few comparisons between groups, then there is less opportunity for false alarms.

For example, consider again the sea mussel data. Group 4 (Far East Russia) and Group 5 (Finland) seem to be different, and it is meaningful to plan a comparison between them because of their geographical difference. If we run a two-group t test, we get $t = 2.53$, $p = 0.028$, which denotes a significant difference. On the other hand, if we run a post-hoc test of all pairwise comparisons, using Tukey's "Honest Significant Difference" correction, then we find that $p = 0.093$, and the difference is not significant. So, do Russia and Finland really differ or not? According to NHST, the answer depends on your intentions: If you intended to compare only those two cases, then they are significantly different, but if you intended to make all pairwise comparisons, then they are not significantly different.

Section 11.4, p. 227, discussed multiple comparisons in NHST in the context of a dichotomous dependent variable. Here we reiterate those ideas in the context of a metric dependent variable.

Suppose that we have two groups: One group is patients treated with a placebo and a second group is patients treated with a totally inactive drug. We measure a metric variable, e.g., body temperature. Because there is no actual difference between the treatments, the underlying distributions of body temperatures are identical for the two groups; we will suppose that they are normally distributed with equal means and equal variances. When we run an experiment, we are collecting a random sample of data from each of the groups. The random samples might show a spuriously large difference between their means, just by chance, despite the fact that on average, in the long run, the groups are identical.

To determine how often the spuriously large differences occur, we can simulate conducting the experiment over and over. For every simulated experiment, we compute the difference of means between the samples from the groups. The difference of sample means is in units of the original measurement scale, e.g., degrees Fahrenheit or degrees Celsius. To get rid of the arbitrary influence of the measurement scale, we standardize the difference of means and call the result the t statistic. Because the true difference between groups is zero, the t value typically will be near zero. Occasionally, by chance, the t value will be

far above or far below zero. The lowest curve in Figure 18.4 shows the probability that the sampled t value falls above the critical t value on the abscissa. For example, the probability that the sampled t value falls above $t_{crit} = 2.23$ is $p(FA) = 0.05$; this is marked by an arrow. In NHST, the decision rule is to reject the null hypothesis if the sample t exceeds a critical value that is selected to keep false alarms to only 5%. Thus, when comparing Group 1 with Group 2, we would reject the null hypothesis if $t > 2.23$, because that would happen only 5% of the time by chance alone.

Now consider an expanded experiment, in which there is a placebo treatment and four distinct drugs, for a total of five treatment groups. According to the null hypothesis, the five treatment groups have identical distributions of body temperatures (normally distributed with equal means and variances). However, because of random sampling in any particular experiment, some treatment samples will have higher or lower mean temperatures than other treatment samples. Suppose that before we collect any data, we plan to compare the placebo group (Group 1) with each of the four drug groups, so we plan four pairwise comparisons. Each of these comparisons might yield a fairly large difference merely by chance, even when there is truly no difference in the underlying distributions. We can determine how often these chance extremes happen by running a Monte Carlo simulation. For a simulated experiment, we randomly sample 6 scores from each of the 5 groups, and compute the t values of each of the 4 comparisons. The simulated experiment is repeated many times. For each candidate t_{crit}, we see what proportion of simulated experiments had a comparison that exceeded that critical value. The middle curve of Figure 18.4 shows the result. Notice that at any given value of t_{crit}, there is now a much higher probability that the simulated experiment will have at least one comparison with t larger than t_{crit}. In particular, to keep the false alarm rate down to 5%, t_{crit} must be about 2.95 instead of 2.33.

If we did not plan only four tests, but instead decided to compare every group with every other group, then we would have even more opportunity for false alarms. With 5 groups, there are 10 pairwise comparisons. If we simulate experiments from equal distributions as before, but this time consider all 10 t values, the probability of false alarm is higher yet, as shown in the right curve of Figure 18.4. The critical value has risen even higher, to approximately 3.43.[1]

Now, suppose we actually run the experiment. We randomly assign 30 people to the 5 groups, 6 people per group. The first group gets the placebo, and the other four groups get the corresponding four drugs. We are careful to make this a double-blind experiment: Neither the subjects nor experimenters know who is getting which treatment. Moreover, no one knows whether any other person is even in the experiment when we collect the data. Our first question is to compare the placebo and the first drug, i.e., group 1 versus group 2. We compute the t statistic for the data from the two groups and find that $t = 2.95$. Do we decide that the two treatments had significantly different effects?

The answer, bizarrely, depends on the intentions of the person we ask. Suppose, for instance, that we handed the data from the first two groups to a research assistant, who is asked to test for a difference between groups. The assistant runs a t-test and finds $t = 2.95$, declaring it to be highly significant because it greatly exceeds the critical value of 2.23 for a two-group t-test. Suppose, on the other hand, that we handed the data from all five groups to a different research assistant, who is asked to compare the first group against each of the other four. This assistant runs a t-test of group 1 versus group 2 and finds $t = 2.95$,

[1] For a discussion of various correction procedures and when to use them, see Figure 5.1 of Maxwell and Delaney (2004). If you must learn NHST methods, this is an excellent resource.

declaring it to be marginally significant because it just squeezes past the critical value of 2.95 for these four planned comparisons. Suppose, on the other hand, that we handed the data from all five groups to a different research assistant, who is told to conduct all pairwise comparisons post-hoc because we have no strong hypotheses about which treatments will have beneficial or detrimental or neutral effects. This assistant runs a t-test of group 1 versus group 2 and finds $t = 2.95$, declaring it to be not significant because it fails to exceed the critical value of 3.43 that is used for post-hoc pairwise comparisons. Notice that regardless of which assistant analyzed the data, the t value for the two groups stayed the same because the data of the two groups stayed the same. Indeed, the data are completely uninfluenced by the intentions of the analyst. So why should the interpretation of the data be influenced by the intentions of the analyst? It shouldn't.

And if you believe that the interpretation should be influenced by the intention of the analyst, how do you determine the intention of the analyst? Did the analyst truly plan only those particular comparisons, or did the analyst really plan others but jettison them once the data were in? Or, did the analyst actually plan fewer comparisons, but realize later that additional comparisons should be made to address theoretical issues? Or, did the analyst actually plan to include two other treatment groups in the study, but then not actually include those groups in the analysis because of administrative errors committed during the data collection? Or what if the experiment was planned by a team of people, some of whom planned some comparisons, and others of whom planned other comparisons? Conclusion: Establishing the true intention of the analyst is not only impossible, it is also impossible.

Multiple comparisons are not a problem in a Bayesian analysis (e.g., Gelman et al., 2009). The posterior distribution is a fixed entity in high-dimensional parameter space, and making comparisons between groups is simply examining the posterior distribution from different perspectives or margins. The posterior does not change when new comparisons come to mind.

The posterior is not immune to spurious coincidences of course. False alarms are mitigated, however, by incorporating prior knowledge into the structure of the model. The estimates of the groups are mutually informative in estimation of higher-level structure, and shrinkage of estimates across groups attenuates false alarms. The attenuation of false alarms is governed by the data, not by unknowable intentions.

18.3 Two group Bayesian ANOVA and the NHST t test

The idea behind a NHST t test is simple: We have two groups, each with a mean. We compute the difference of the means, and standardize that difference relative to the standard deviation of the scores within the groups. The resulting standardized difference is called the t value. We want to know whether the observed t value is significantly different from zero, so we compare the t value to a sampling distribution of t values (Gosset, 1908). The sampling distribution assumes that the intention of the researcher was to stop when there were exactly N_1 values observed for the first group, and exactly N_2 values observed for the second group.

The t test is a special case of NHST ANOVA when there are only two groups. More specifically, when the two groups are assumed to have equal variances in the underlying population, then the t value squared equals the F value in two-group ANOVA. (And what's an F value, you may ask? The F value is the summary statistic used in NHST ANOVA to express how much the groups differ from each other. It's the ratio of the variance between

group means, relative to the variance within groups.)

In typical applications of BANOVA, the prior on the between-group variance is only mildly informed. In this case, a BANOVA on two groups imposes little shrinkage on the group estimates because there are so few groups. It is only when several groups "gang up" that they strongly inform the estimate of the variation between groups, and therefore constrain the estimates of other groups. When the prior on within-group variance is also vague, the results of a two-group BANOVA closely agree with the results of an NHST t-test. Exercise 18.1 has you explore this correspondence.

18.4 R code

18.4.1 Bayesian oneway ANOVA

(ANOVAonewayBRugs.R)

```
graphics.off()
rm(list=ls(all=TRUE))
fnroot = "ANOVAonewayBrugs"
library(BRugs)              # Kruschke, J. K. (2010). Doing Bayesian dat      a analysis:
                              # A Tutorial with R and BUGS. Academic Press / Elsevier.
#------------------------------------------------      ----------------------------
# THE MODEL.

modelstring = "
# BUGS model specification begins here...
model {
  for ( i in 1:Ntotal ) {
    y[i] ~ dnorm( mu[i] , tau )
    mu[i] <- a0 + a[x[i]]
  }
  #
  tau <- pow( sigma , -2 )
  sigma ~ dunif(0,10) # y values are assumed to be standardized
  #
  a0 ~ dnorm(0,0.001) # y values are assumed to be standardized
  #
  for ( j in 1:NxLvl ) { a[j] ~ dnorm( 0.0 , atau ) }
  atau <- 1 / pow( aSD , 2 )
  aSD <- abs( aSDunabs ) + .1
  aSDunabs ~ dt( 0 , 0.001 , 2 )
}
# ... end BUGS model specification
" # close quote for modelstring
# Write model to a file, and send to BUGS:
writeLines(modelstring,con="model.txt")
modelCheck( "model.txt" )

#------------------------------------------------      ----------------------------
# THE DATA.

# Specify data source:
dataSource = c( "McDonaldSK1991" , "SolariLS2008" , "Rando  m" )[1]
# Load the data:

if ( dataSource == "McDonaldSK1991" ) {
  fnroot = paste( fnroot , dataSource , sep="" )
```

```
  datarecord = read.table( "McDonaldSK1991data.txt", head      er=T ,
                           colClasses=c("factor","numeric") )
  y = as.numeric(datarecord$Size)
  Ntotal = length(datarecord$Size)
  x = as.numeric(datarecord$Group)
  xnames = levels(datarecord$Group)
  NxLvl = length(unique(datarecord$Group))
  contrastList = list( BIGvSMALL = c(-1/3,-1/3,1/2,-1/3,1/          2) ,
                       ORE1vORE2 = c(1,-1,0,0,0) ,
                       ALAvORE = c(-1/2,-1/2,1,0,0) ,
                       NPACvORE = c(-1/2,-1/2,1/2,1/2,0) ,
                       USAvRUS = c(1/3,1/3,1/3,-1,0) ,
                       FINvPAC = c(-1/4,-1/4,-1/4,-1/4,1) ,
                       ENGvOTH = c(1/3,1/3,1/3,-1/2,-1/2) ,
                       FINvRUS = c(0,0,0,-1,1) )
}

if ( dataSource == "SolariLS2008" ) {
  fnroot = paste( fnroot , dataSource , sep="" )
  datarecord = read.table("SolariLS2008data.txt", header         =T ,
                           colClasses=c("factor","numeric") )
  y = as.numeric(datarecord$Acid)
  Ntotal = length(datarecord$Acid)
  x = as.numeric(datarecord$Type)
  xnames = levels(datarecord$Type)
  NxLvl = length(unique(datarecord$Type))
  contrastList = list( G3vOTHER = c(-1/8,-1/8,1,-1/8,-1/8,          -1/8,-1/8,-1/8,-1/8) )
}

if ( dataSource == "Random" ) {
  fnroot = paste( fnroot , dataSource , sep="" )
  #set.seed(47405)
  ysdtrue = 4.0
  a0true = 100
  atrue = c( 2 , -2 ) # sum to zero
  npercell = 8
  datarecord = matrix( 0, ncol=2 , nrow=length(atrue)*nperc        ell )
  colnames(datarecord) = c("y","x")
  rowidx = 0
  for ( xidx in 1:length(atrue) ) {
    for ( subjidx in 1:npercell ) {
      rowidx = rowidx + 1
      datarecord[rowidx,"x"] = xidx
      datarecord[rowidx,"y"] = ( a0true + atrue[xidx] + rnorm(1,          0,ysdtrue) )
    }
  }
  datarecord = data.frame( y=datarecord[,"y"] , x=as.facto          r(datarecord[,"x"]) )
  y = as.numeric(datarecord$y)
  Ntotal = length(y)
  x = as.numeric(datarecord$x)
  xnames = levels(datarecord$x)
  NxLvl = length(unique(x))
  # Construct list of all pairwise comparisons, to compare wit        h NHST TukeyHSD:
  contrastList = NULL
  for ( g1idx in 1:(NxLvl-1) ) {
    for ( g2idx in (g1idx+1):NxLvl ) {
      cmpVec = rep(0,NxLvl)
      cmpVec[g1idx] = -1
      cmpVec[g2idx] = 1
```

```
101         contrastList = c( contrastList , list( cmpVec ) )
102       }
103     }
104   }
105
106   # Specify the data in a form that is compatible with BRugs mode     l, as a list:
107   ySDorig = sd(y)
108   yMorig = mean(y)
109   z = ( y - yMorig ) / ySDorig
110   datalist = list(
111     y = z ,
112     x = x ,
113     Ntotal = Ntotal ,
114     NxLvl = NxLvl
115   )
116   # Get the data into BRugs:
117   modelData( bugsData( datalist ) )
118
119   #------------------------------------------------                    ----------------------------
120   # INTIALIZE THE CHAINS.
121
122   # Autocorrelation within chains is large, so use several cha          ins to reduce
123   # degree of thinning. But we still have to burn-in all the chai            ns, which takes
124   # more time with more chains (on serial CPUs).
125   nchain = 5
126   modelCompile( numChains = nchain )
127
128   if ( F ) {
129       modelGenInits() # often won't work for diffuse prior
130   } else {
131     #   initialization based on data
132     theData = data.frame( y=datalist$y , x=factor(x,labels=x           names) )
133     a0 = mean( theData$y )
134     a = aggregate( theData$y , list( theData$x ) , mean )[,2] - a0
135     ssw = aggregate( theData$y , list( theData$x ) ,
136                           function(x){var(x)*(length(x)-1)} )[,2]
137     sp = sqrt( sum( ssw ) / length( theData$y ) )
138     genInitList <- function() {
139       return(
140           list(
141                 a0 = a0 ,
142                 a = a ,
143                 sigma = sp ,
144                 aSDunabs = sd(a)
145           )
146       )
147     }
148     for ( chainIdx in 1 : nchain ) {
149       modelInits( bugsInits( genInitList ) )
150     }
151   }
152
153   #------------------------------------------------                    ----------------------------
154   # RUN THE CHAINS
155
156   # burn in
157   BurnInSteps = 10000
158   modelUpdate( BurnInSteps )
159   # actual samples
```

```
160 samplesSet( c( "a0" ,  "a" , "sigma" , "aSD" ) )
161 stepsPerChain = ceiling(5000/nchain)
162 thinStep = 750
163 modelUpdate( stepsPerChain , thin=thinStep )
164
165 #-------------------------------------------------                    ----------------------------
166 # EXAMINE THE RESULTS
167
168 source("plotChains.R")
169 source("plotPost.R")
170
171 checkConvergence = T
172 if ( checkConvergence ) {
173     sumInfo = plotChains( "a0" , saveplots=T , filenameroot=fn        root )
174     sumInfo = plotChains( "a" , saveplots=T , filenameroot=fnr        oot )
175     sumInfo = plotChains( "sigma" , saveplots=T , filenameroot        =fnroot )
176     sumInfo = plotChains( "aSD" , saveplots=T , filenameroot=f        nroot )
177 }
178
179 # Extract and plot the SDs:
180 sigmaSample = samplesSample("sigma")
181 aSDSample = samplesSample("aSD")
182 windows()
183 layout( matrix(1:2,nrow=2) )
184 par( mar=c(3,1,2.5,0) , mgp=c(2,0.7,0) )
185 plotPost( sigmaSample , xlab="sigma" , main="Cell SD" , bre      aks=30 )
186 plotPost( aSDSample , xlab="aSD" , main="a SD" , breaks=30 )
187 dev.copy2eps(file=paste(fnroot,"SD.eps",sep=""))
188
189 # Extract a values:
190 a0Sample = samplesSample( "a0" )
191 chainLength = length(a0Sample)
192 aSample = array( 0 , dim=c( datalist$NxLvl , chainLength ) )
193 for ( xidx in 1:datalist$NxLvl ) {
194     aSample[xidx,] = samplesSample( paste("a[",xidx,"]",se          p="") )
195 }
196
197 # Convert to zero-centered b values:
198 mSample = array( 0, dim=c( datalist$NxLvl , chainLength ) )
199 for ( stepIdx in 1:chainLength ) {
200       mSample[,stepIdx ] = ( a0Sample[stepIdx] + aSample[,stepI      dx] )
201 }
202 b0Sample = apply( mSample , 2 , mean )
203 bSample = mSample - matrix(rep( b0Sample ,NxLvl),nrow=NxL vl,byrow=T)
204 # Convert from standardized b values to original scale b valu       es:
205 b0Sample = b0Sample * ySDorig + yMorig
206 bSample = bSample * ySDorig
207
208 # Plot b values:
209 windows(datalist$NxLvl*2.75,2.5)
210 layout( matrix( 1:datalist$NxLvl , nrow=1 ) )
211 par( mar=c(3,1,2.5,0) , mgp=c(2,0.7,0) )
212 for ( xidx in 1:datalist$NxLvl ) {
213       plotPost( bSample[xidx,] , breaks=30 ,
214                 xlab=bquote(beta*1[.(xidx)]) ,
215                 main=paste("x:",xnames[xidx])   )
216 }
217 dev.copy2eps(file=paste(fnroot,"b.eps",sep=""))
218
```

```
# Display contrast analyses
nContrasts = length( contrastList )
if ( nContrasts > 0 ) {
    nPlotPerRow = 5
    nPlotRow = ceiling(nContrasts/nPlotPerRow)
    nPlotCol = ceiling(nContrasts/nPlotRow)
    windows(3.75*nPlotCol,2.5*nPlotRow)
    layout( matrix(1:(nPlotRow*nPlotCol),nrow=nPlotRow,n col=nPlotCol,byrow=T) )
    par( mar=c(4,0.5,2.5,0.5) , mgp=c(2,0.7,0) )
    for ( cIdx in 1:nContrasts ) {
        contrast = matrix( contrastList[[cIdx]],nrow=1) # make it a row matrix
        incIdx = contrast!=0
        histInfo = plotPost( contrast %*% bSample , compVal=0 , brea ks=30 ,
                xlab=paste( round(contrast[incIdx],2) , xnames[incIdx] ,
                        c(rep("+",sum(incIdx)-1),"") , collapse=" " ) ,
                cex.lab = 1.0 ,
                main=paste( "X Contrast:", names(contrastList)[cIdx] ) )
    }
    dev.copy2eps(file=paste(fnroot,"xContrasts.eps",sep ="")) }

#=============================================================================
# Do NHST ANOVA and t tests:

theData = data.frame( y=y , x=factor(x,labels=xnames) )
aovresult = aov( y ~ x , data = theData ) # NHST ANOVA
cat("\n---------- ----------\n\n")
print( summary( aovresult ) )
cat("\n---------- ----------\n\n")
print( model.tables( aovresult , "means" ) , digits=4 )
windows()
boxplot( y ~ x , data = theData )
cat("\n---------- ----------\n\n")
print( TukeyHSD( aovresult , "x" , ordered = FALSE ) )
windows()
plot( TukeyHSD( aovresult , "x" ) )
if ( T ) {
  for ( xIdx1 in 1:(NxLvl-1) ) {
    for ( xIdx2 in (xIdx1+1):NxLvl ) {
      cat("\n---------- ----------\n\n")
      cat( "xIdx1 = " , xIdx1 , ", xIdx2 = " , xIdx2 ,
           ", M2-M1 = " , mean(y[x==xIdx2])-mean(y[x==xIdx1]) , "\n" )
      print( t.test( y[x==xIdx2] , y[x==xIdx1] , var.equal=T ) ) # t test
    }
  }
}
cat("\n---------- ----------\n\n")

#=============================================================================
```

18.5 Exercises

Exercise 18.1. [Purpose: Notice that Bayesian ANOVA with two groups tends to agree with an NHST t test.] The BRugs program of Section 18.4.1 (ANOVAonewayBRugs.) allows you to specify random data. It executes a Bayesian ANOVA, and at the end of the program it also conducts an NHST ANOVA and t tests (using R's aov and t.test functions). Run the program

ten times with different random data by commenting out the `set.seed` command. Specify `sdtrue = 4.0 , atrue = c(2,-2)` (which implies two groups because there are two deflections), and `nPerCell = 8` . For each run, record, by hand, (i) how much of the posterior difference between means falls on one side of zero (see the `plotPost` histogram with the main title "X Contrast" and x-axis labeled " 1 1 + 1 2"), (ii) whether the 95% HDI excludes zero, and (iii) the confidence interval and p-value of the NHST t-test. Do the t-test and the BANOVA usually agree in their decisions about whether the group means are different?

Exercise 18.2. [Purpose: Influence of the prior in Bayesian ANOVA.] In the model section of the BRugs program of Section 18.4.1 (`ANOVAonewayBRugs.R`), and correspondingly in the diagram of Figure 18.1, there are several constants that determine the prior. These constants include the mean value of the baseline (M_0 in the diagram), the precision on the baseline (T_0 in the diagram), the precision of the folded-t distribution (T_t in the diagram), and the upper value of the uniform distribution on $_y$ ($H_{_y}$ in the diagram). Because the data are standardized, M_0 should be set at zero, and T_0 can be modest (not terribly small). $H_{_y}$ also can be set to a modest value because the data are standardized. But what about the precision of the folded-t distribution, T_t? This constant modulates the degree of shrinkage: A large value of T_t indicates prior knowledge that the groups do not differ much, and imposes a high degree of shrinkage that must be overcome by the data.

Run the program on the mussel data using a small value of T_t, such as 1.0E-6, and a large value of T_t, such as 1000. Are the results very different? Discuss which prior value might be appropriate.

Exercise 18.3. [Purpose: Bayesian ANOVA without assuming equal variances.] Modify the program in Section 18.4.1 (`ANOVAonewayBRugs.R`) so that it allows a different variance for each group, with the different variances coming from a hyperdistribution that has its precision informed by the data. In other words, instead of assuming the same $_y$ (= $1=\,^2_y$) for all the levels of x, we allow each group to have its own variance. Denote the precision of j^{th} group as $_j$, analogous to the deflection $_j$. Just as the group deflections are assumed to come from a higher-level distribution, we will assume that the group SD's come from a higher-level distribution. Because SD's must be non-negative, use a gamma density for the higher-level distribution. The gamma distribution has two parameters for which you need to establish a prior. See the right side of Figure 16.11, p. 356, for guidance. Corresponding code is offered in a hint, below. Run the program on the mussel muscle data. Are the conclusions about the group means any different than when assuming equal variances across groups?

Hint regarding the conclusion: The posteriors on the group means are only a little different in this case, because the groups variances are roughly the same. But, because the group variances are less constrained when they are allowed to be different, they are less certain. Therefore the group means are a little less certain, and therefore the differences of means are a little less certain.

Programming hints: Here are some code snippets, showing model specification and chain initialization. (`ANOVAonewayNonhomogVarBrugs.R`)

```
11 model {
12   for ( i in 1:Ntotal ) {
13     y[i] ~ dnorm( mu[i] , tau[x[i]] )
14     mu[i] <- a0 + a[x[i]]
15   }
16   a0 ~ dnorm(0,0.001)
17   for ( j in 1:NxLvl ) {
```

```
        a[j] ~ dnorm( 0.0 , atau )
        tau[j] ~ dgamma( sG , rG )
    }
    sG <- pow(m,2)/pow(d,2)
    rG <- m/pow(d,2)
    m ~ dgamma(1,1)
    d ~ dgamma(1,1)
    atau <- 1 / pow( aSD , 2 )
    aSD <- abs( aSDunabs ) + .1
    aSDunabs ~ dt( 0 , 0.001 , 2 )
}
```

(ANOVAonewayNonhomogvarBrugs.R)

```
    #   initialization based on data
    theData = data.frame( y=datalist$y , x=factor(x,labels=x           names) )
    a0 = mean( theData$y )
    a = aggregate( theData$y , list( theData$x ) , mean )[,2] - a0
    tau = 1/(aggregate( theData$y , list( theData$x ) , sd )[,2])           ^2
    genInitList <- function() {
      return(
          list(
              a0 = a0 ,
              a = a ,
              tau = tau ,
              m = mean( tau ) ,
              d = sd( tau ) ,
              aSDunabs = sd(a)
          )
      )
    }
    for ( chainIdx in 1 : nchain ) {
      modelInits( bugsInits( genInitList ) )
    }
```

Chapter 19

Metric Predicted Variable with Multiple Nominal Predictors

Contents

> Sometimes I wonder just how it could be, that
> Factors aligned so you'd end up with me.
> All of the priors made everyone think, that
> Our interaction was destined to shrink.

In this chapter we consider situations with a metric predicted variable and multiple nominal predictor variables. For example, we might want to predict income (a metric variable) on the basis of political party affiliation (a nominal variable) and ethnicity (another nominal variable). Or, we may want to predict response time (a metric variable) on the basis of hand used for the response (a nominal value: dominant hand or non-dominant hand) and modality of stimulus (another nominal value: visual, auditory, or tactile). These situations are modeled by the cell in the first row and last column of Table 14.1, p. 312.

In traditional NHST, this situation is known as multifactor ANOVA. We use the same underlying model, but without reference to sampling distributions, and instead with hierarchical priors that provide additional structural constraints. Multifactor ANOVA is a straightforward extension of the model in the previous chapter, but including the a new

concept of interaction between nominal variables. Just as simple regression considered interaction of metric predictors, multifactor ANOVA considers interaction of nominal predictors.

19.1 Bayesian multi-factor ANOVA

Recall from the previous chapter that in oneway ANOVA we described the effect of each level of the predictor as a deflection away from an overall baseline, where the baseline is the central tendency across all levels of the predictor. In multifactor ANOVA, the same idea applies to two or more predictors, and the deflections due to each predictor are added. We'll use notation analogous to the previous chapter, but with extra subscripts to indicate the different predictors, just as we used in multiple regression for continuous predictors.

The mathematical notation was introduced as a case of the generalized linear model in Section 14.1.6.2, p. 302. Suppose we have two nominal predictors, cleverly denoted x_1 and x_2. These predictor vectors can only take on values like $\langle 1,0,0,\ldots\rangle$, $\langle 0,1,0,\ldots\rangle$, and so on, with the j^{th} component having the value 1 when the predictor has its j^{th} nominal level.

When the effects of the two predictors are additive, the predicted trend is as follows:

$$\begin{aligned} y &= \beta_0 + \vec{\beta}_1 \vec{x}_1 + \vec{\beta}_2 \vec{x}_2 \\ &= \beta_0 + \sum_{j=1}^{J_1} \beta_{1;j} x_{1;j} + \sum_{k=1}^{J_2} \beta_{2;k} x_{2;k} \end{aligned}$$

To make the parameter values unique, we include the constraints

$$\sum_{j=1}^{J_1} \beta_{1;j} = 0 \quad \text{and} \quad \sum_{j=k}^{J_2} \beta_{2;k} = 0$$

Those equations repeat Equations 14.7 and 14.8. In words, the value β_0 establishes the overall baseline from which the predictors indicate deflections. When predictor x_1 has value $x_{1;j}$, a deflection of $\beta_{1;j}$ is added to the baseline, and when predictor x_2 has value $x_{2;k}$, a deflection of $\beta_{2;k}$ is also added to the baseline. The deflections may be negative indeed, across all levels of the predictors, the constraints demand as much negative deflection as positive deflection, so that the deflections sum to zero for each predictor.

19.1.1 Interaction of nominal predictors

The effect of two predictors may be non-additive, in which case we say that there is an "interaction" of the predictors. For example, if a flame is put under a hot-air balloon, its buoyancy will increase. And if hydrogen is added to a balloon, its buoyancy will increase. But if hydrogen and flame are added to a balloon, there is a non-additive interaction, such that buoyancy is not increased.

Figure 19.1 displays a simple interaction. Both predictors have only two levels. The abscissa groups the two levels of predictor x_1, and the shading of the bars indicates the two levels of predictor x_2. All three panels of Figure 19.1 show the same data, but the nature of the interaction is highlighted differently in each panel.

In the left panel of Figure 19.1, the dashed parallelogram indicates the best additive model for the data. The dashed lines indicate the average height when the levels of the

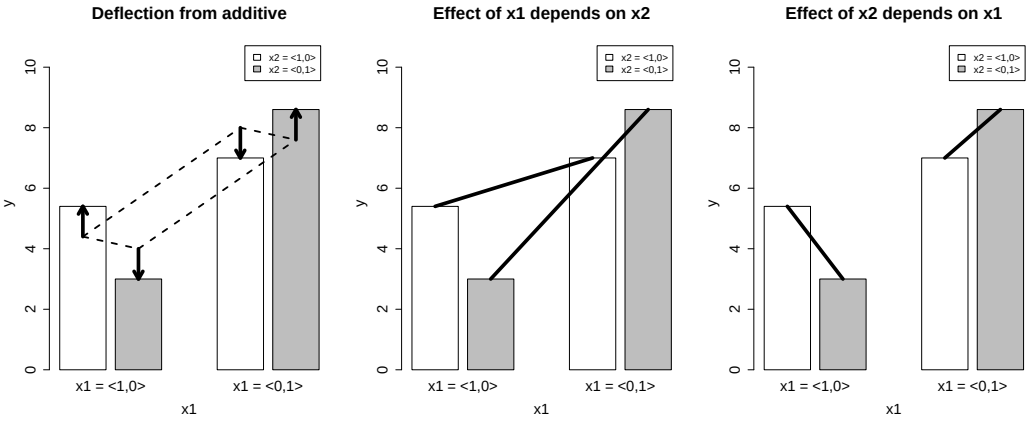

Figure 19.1: An interaction of nominal variables x_1 and x_2, parsed three ways. The left panel emphasizes that the interaction involves a non-additive, torsion-like deflection away from the additive model, as indicated by the arrows. The middle panel shows the same data, with lines that emphasize that the effect of x_1 depends on the value of x_2. The right panel again shows the same data, but with lines that emphasize that the effect of x_2 depends on the value of x_1.

predictors change. The vertical arrows highlight the non-additive deflections, away from the additive average, that constitute the interaction. Notice that the arrows sum to zero across each edge of the parallelogram. Thus, the interaction components do not change the average deflections of each predictor.

The middle and right panels of Figure 19.1 highlight different interpretations of the interaction. The middle panel shows that the effect of x_1, i.e., the amount that y changes when x_1 changes, depends on the level of x_2: When $x_2 = \langle 1,0 \rangle$, there is only a small change in y when x_1 changes, but when $x_2 = \langle 0,1 \rangle$, there is only a larger change in y when x_1 changes. The right panel makes the same point but with the role of the predictors reversed: When $x_1 = \langle 1,0 \rangle$, the effect of x_2 is to decrease y, but when $x_1 = \langle 0,1 \rangle$, the effect of x_2 is to increase y.

The average deflection from baseline due to a predictor is called a main effect of the predictor. The main effects of the predictors correspond to the dashed lines in the left panel of Figure 19.1. When there is non-additive interaction between predictors, the effect of one predictor depends on the level of the other predictor. The deflection from baseline for a predictor, at a fixed level of the other predictor, is called a simple effect of the predictor at the level of the other predictor. When there is interaction, the simple effects do not equal the main effect.

It may be edifying to compare Figure 19.1, which shows interaction of nominal predictors, with Figure 17.8, p. 384, which shows interaction of metric predictors. The essential notion of interaction is the same in both cases: Interaction is the non-additive portion of the prediction, and interaction means that the effect of one predictor depends on the level of the other predictor.

The mathematical formalism for non-additive interactions was introduced in Section 14.1.6.3, p. 303, as is repeated here. The non-additive components, indicated by the vertical arrows in Figure 19.1, are denoted $a_{1:2;j;k}$, which means the interaction of predictors 1 and 2 (denoted 1:2) at level j of predictor 1 and level k of predictor 2. The formal

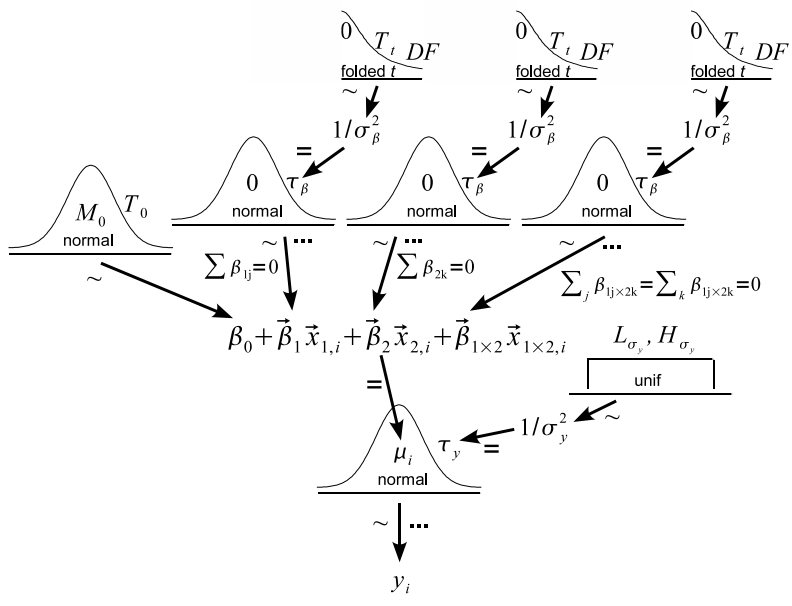

Figure 19.2: Hierarchical dependencies for model of two-way Bayesian ANOVA. Compare with Figure 18.1.

expression merely expands the additive model with by including the interaction. Recall from Equations 14.9 and 14.10 that the model with interaction can be written as

$$\begin{aligned} y &= \beta_0 + \vec{\beta}_1 \vec{x}_1 + \vec{\beta}_2 \vec{x}_2 + \vec{\beta}_{1\times2} \vec{x}_{1\times2} \\ &= \beta_0 + \sum_{j=1}^{X_1} \beta_{1;j} x_{1;j} + \sum_{k=1}^{X_2} \beta_{2;k} x_{2;k} + \sum_{j=1}^{X_1} \sum_{k=1}^{X_2} \beta_{1\times2;j,k} x_{1\times2;j,k} \end{aligned}$$

with the constraints

$$\sum_{j=1}^{X_1} \beta_{1;j} = 0 \quad \text{and} \quad \sum_{k=1}^{X_2} \beta_{2;k} = 0 \quad \text{and} \quad \sum_{j=1}^{X_1} \beta_{1\times2;j,k} = 0 \;\forall k \quad \text{and} \quad \sum_{k=1}^{X_2} \beta_{1\times2;j,k} = 0 \;\forall j$$

In those last equations, the symbol "$\forall$" means "for all". In words, the last two equations simply mean that the interaction deviations sum to zero across every level of the two predictors. A graphic example of this was shown in the left panel of Figure 19.1, where it can be seen that the heights of the arrows sum to zero along every edge of the parallelogram.

Our goal is to estimate the additive and interactive deviations, based on the observed data. It is important to understand that the observed data are not the bars in Figure 19.1; instead, the data are swarms of points at various heights that cluster around the heights of the bars. The bars represent the central tendency of the data at each combination of the predictors. Thus, what the equations above actually predict is the central tendency at each combination of predictors, and the data are typically modeled as being randomly distributed around .

19.1.2 The hierarchical prior

The complete generative model of the data is shown in Figure 19.2. It might look daunting, but it really is merely the diagram for oneway ANOVA, in Figure 18.1, with the hyperprior replicated for each predictor and interaction.

The lowest level of Figure 19.2 indicates that the observed data points, y_i, are distributed normally around the predicted value, μ. Moving upward in the diagram, the arrow impinging upon μ_i indicates that the predicted value is baseline plus additive deviation due to each predictor plus interactive deviation due to the combination of predictors. The upper levels of the diagram indicate prior structural assumptions about the deviations. We assume that the deviations produced by a predictor are centered at zero, and we allow the variance (i.e., precision) of the deviations to be estimated from the data. That is, if most of the deviations are small, the estimated variance is small, and the hyperdistribution creates shrinkage in the estimates of other deviations.

A key conceptual aspect of the hyperdistributions is that they apply separately to the different predictors and interactions. In other words, there is not just one hyperdistribution that governs all deviations for all predictors and interactions. This division of generative structure reflects a prior assumption that the magnitude of the effect of one predictor might not be very informative regarding the magnitude of the effect of a different predictor. But, within a predictor, the magnitude of deviation produced by one level may inform the magnitude of deviation produced by other levels of that same predictor.[1]

As was assumed in the case of oneway ANOVA, we will assume homogeneity of variance: The variability of the observed data is the same within each combination of predictors. This is indicated in Figure 19.2 by the single parameter σ_y that is used in likelihood function, regardless of the values of the predictors. As before, there are two reasons for this assumption. First, the assumption is a natural simplification in multiple regression on metric predictors, and ANOVA can be construed as a special case of multiple regression. Second, the assumption of equal variances is made in NHST ANOVA, and we will also make it here in BANOVA to facilitate comparing across the techniques. But there is no requirement in BANOVA to assume equal variances. If the the situation suggests that different levels of the predictors produce radically different variances in the data, then the hierarchical prior can allow different variances.

19.1.3 An example in R and BUGS

Figure 19.3 shows the mean annual salaries of faculty in four departments at three levels of seniority. The four departments are business finance, cognitive psychology, chemistry, and theater. These departments are the nominal levels of a predictor denoted x_1. The three levels of seniority are full professor, associate professor, and assistant professor. Assistant professors are usually within five years after completing their doctoral or post-doctoral studies. Associate professors are usually within about ten years of their doctoral or post-doctoral studies. Full professors are anywhere from 10 to 40 years post graduate school. Although seniority could, and perhaps should, be treated as an ordinal variable, we will treat it as a nominal predictor, denoted x_2. A glance at the means suggests that there are effects of department and of seniority. There appears also to be an interaction, meaning that the change in salary due to seniority depends on department. Our goal is to estimate the baseline salary, the main effect of department membership, the main effect of seniority, and the interaction of department and seniority.

The display of the means in Figure 19.3 obscures the fact that different combinations of

[1] By analogy to multiple regression, if there are many predictors included in a model, it is reasonable in principle to include a higher-level distribution across predictors such that the estimated variance of one predictor informs the estimated variance of another predictor. This would be especially useful if the application includes many nominal predictors, each with many levels. Such applications are rare.

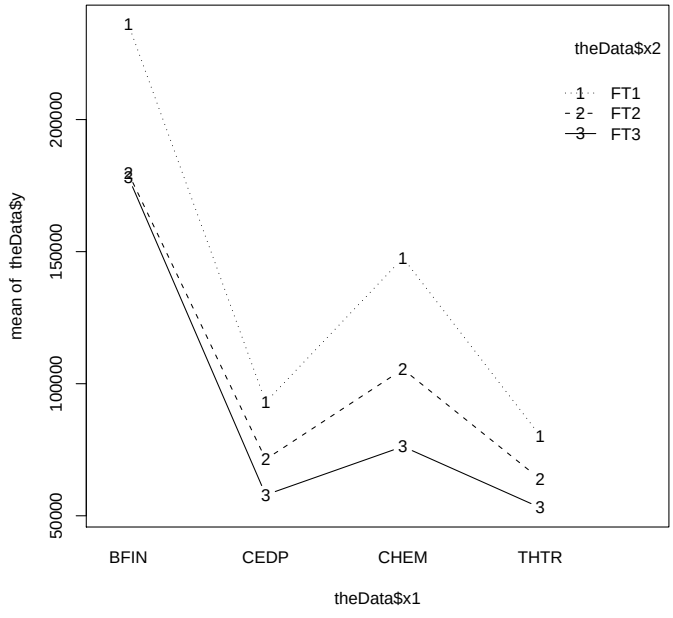

Figure 19.3: Mean annual salaries of faculty in four departments at three levels of seniority.

department and seniority had different numbers of data points. In other words, the number of associate professors in business nance was not necessarily equal to the number of full professors in theater. In traditional NHST ANOVA, this sort of "unbalanced" design can cause serious computational difficulties (e.g., Maxwell & Delaney, 2004, pp. 320–343). But Bayesian ANOVA has no problem with unbalanced designs.

The model of Figure 19.2 was implemented in R and BRugs and is listed in Section 19.3.1 (ANOVAtwowayBRugs.R). Several tricks for running the model in BUGS are described in that section, before the program listing. The results, however, are much like the oneway ANOVA model of the previous chapter.

The results are shown in Figure 19.4. (The means and HDIs are displayed with only three signi cant digits, but more precise values can be obtained directly from the program.) The top left histogram shows that the baseline for these four departments is 111,381. Notice, however, that most of the data fall below the baseline because the overall data are skewed by the much higher salaries in one department. For salaries in the department of chemistry, the fourth histogram in the top row indicates that 2,164 should be subtracted from the baseline. For salaries of assistant professors, the rst histogram in the bottom row indicates that 20,100 should be subtracted from the baseline. Thus, for assistant professors in the department of chemistry, the linearly-predicted salary is $111{,}381 - 2{,}164 - 20{,}100 = 89{,}117$. But there is a notable non-linear interaction component for that combination: The fourth histogram of the bottom row shows that 10,938 must be subtracted from the linear combination to get the mean salary for that combination, namely, 78,179.

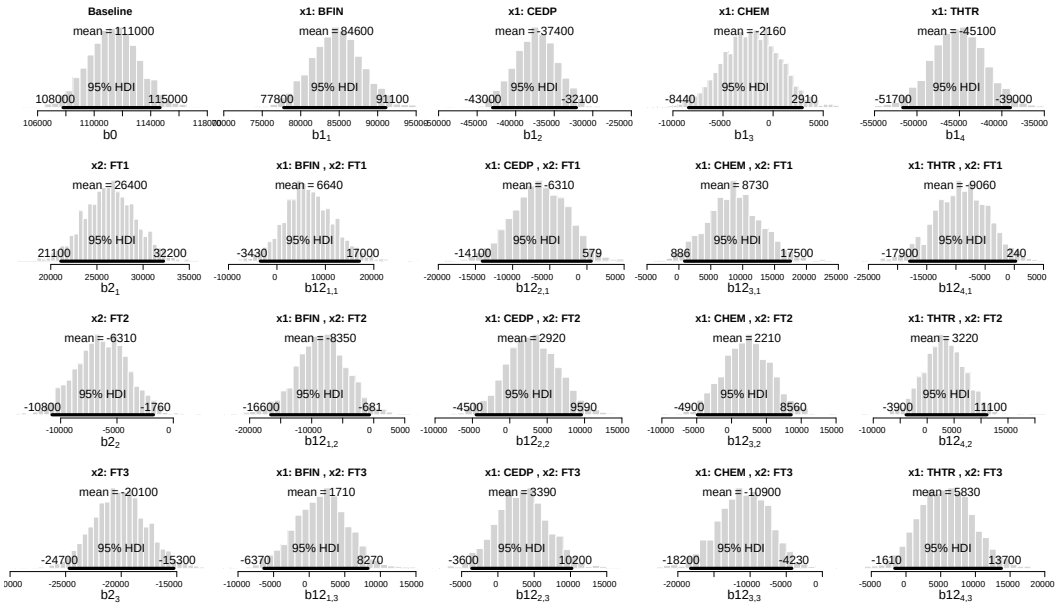

Figure 19.4: Posterior distribution for data in Figure 19.3. Baseline (β_0) is shown in upper left. Remainder of top row is main effect of x_1 (department). Remainder of left column is main effect of x_2 (seniority). Remaining cells show the interaction effects.

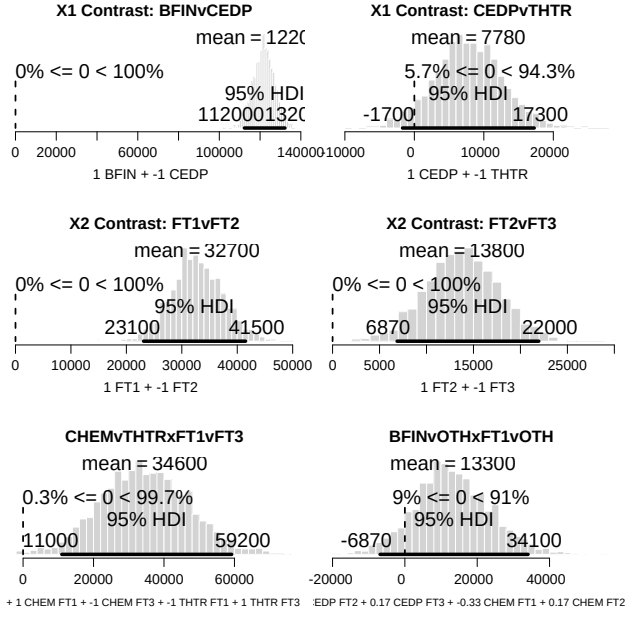

Figure 19.5: Selected contrasts for posterior in Figure 19.4.

19.1.4 Interpreting the posterior

In most applications, we are interested not only in estimating the main effects for each group, but we are also interested in deciding whether two groups are credibly different. Just as we compared groups in oneway ANOVA in the previous chapter, we can compare groups in multifactor ANOVA.

The top and middle rows of Figure 19.5 show selected contrasts of levels of the main effects. We may ask whether there is a credible difference in salaries, on average, between business finance (BFIN) and counseling and educational psychology (CEDP). The top left histogram indicates that the average difference is about $122,000, and the 95% HDI falls far from zero. We may also ask whether there is a credible difference in salaries, on average, between CEDP and theater (THTR). The top right histogram indicates that the average difference is about $7,780, but the 95% HDI spans zero, which indicates that the uncertainty in the estimated difference is fairly large relative to the estimated difference itself. The middle row of Figure 19.5 shows contrasts regarding levels of seniority: There is a credible difference between full professors (FT1) and associate professors (FT2), and between FT2 and assistant professors (FT3).

It is important to understand that the main effects of department and seniority are average effects, when the other factors are collapsed. For example, the contrast between FT2 and FT3 (middle row, right panel of Figure 19.5) is the average difference between FT2 and FT3, collapsed across all departments. But if you look at the data in Figure 19.3, you can see that the difference between FT2 and FT3 is not the same in every department. There is a fairly large difference in CHEM, but a very small difference in BFIN. The effect of changing from FT2 to FT3 depends on the department, which means that there is interaction.

Main effects must be interpreted and described cautiously when there are interactions. It would be a mistake to say that "the" difference between FT2 and FT3 is 13,800. Instead, that is the average difference across departments. The actual difference within any particular department might be quite different. Similarly, it would be a mistake to say that "the" effect of FT3 is to subtract 20,100 from baseline, because the effect of seniority interacts with department.

19.1.4.1 Metric predictors and ANCOVA

Consider again the salary data in Figure 19.3. You can see that mean salary for FT1's in Chemistry is much higher than in Theater. This difference might be attributable solely to being in one department or the other. But the difference might also be attributable to some other factor, such as years on the job. In other words, FT1's in Chemistry might happen to have been employed for decades, while the FT1's in Theater might happen to be relatively young. If we had the age of each employee, or better, the number of years that the employee had been at the current level of seniority, then we could include that information as an additional predictor of salary. We could then assess whether department membership contributed any predictiveness beyond number of years on the job.

When a nominal predictor, such as department membership, is combined with a metric predictor, such as years on the job, the model is sometimes referred to as analysis of covariance, or ANCOVA. The metric predictor is the "covariate".

Programming ANCOVA in BUGS is a trivial combination of the models we've used for linear regression and ANOVA. Denote the nominal group membership for individual i as `xNom[i]`, and denote the metric covariate value as `xMet[i]`. Then the core of the BUGS

model speci cation is

```
mu[i] <- a0 + a[ xNom[i] ] + bMet * xMet[i]
y[i]       dnorm( mu[i] , tau )
```

where a[] is the de ection of each group from baseline, and bMet is the regression coecient on the covariate.

19.1.4.2 Interaction contrasts

Just as we can ask whether dierences among particular levels of predictors are credible, we can ask whether interactions among particular combinations of predictors are credible. Consider again the data in Figure 19.3. The dience between full professors (FT1) and assistant professors (FT3) appears to be large in the chemistry partment (CHEM) but smaller in the theater department (THTR). Is the simple ect of seniority bigger in chemistry than it is in theater? In other words, is (CHEM.FT1 CHEM.FT3) (THTR.FT1 THTR.FT3) credibly non-zero?

This sort of di erence of di erences is called an interaction contrast In general, an interaction contrast is constructed by taking any set of contrast coecients on x_1, and any set of contrast coecients on x_2, and computing their outer product. The outer product was described in Section 8.8.1 (BernTwoGrid.R), p. 144. Formally, the outer product of two vectors is denoted by the symbol "$\otimes$". To provide an example of an interaction contrast as an outer product of main ect contrasts, we will re-cast the one we are presently considering, namely (CHEM.FT1 CHEM.FT3) (THTR.FT1 THTR.FT3), in generic notation. Notice that CHEM is level 3 of predictor 1, hence can be written as $x_{1;3}$. Writing the other components in the same fashion, the interaction contrast is $(x_{1;3} : x_{2;1} - x_{1;3} : x_{2;3}) - (x_{1;4} : x_{2;1} - x_{1;4} : x_{2;3})$. That can be algebraically re-arranged to highlight the coecients on the particular combinations:

$$(+1)\, x_{1;3} : x_{2;1} + (-1)\, x_{1;3} : x_{2;3} + (-1)\, x_{1;4} : x_{2;1} + (+1)\, x_{1;4} : x_{2;3}$$

Those highlighted coecients can be obtained as the outer product of main ect contrasts, namely the contrast $c_1 = h0; 0; +1; -1i$, which expresses CHEM minus THTR, and the contrast $c_2 = h+1; 0; -1i$, which expresses FT1 minus FT3:

$$c_1 \otimes c_2 = \begin{matrix} & x_{1;1} & x_{1;2} & x_{1;3} & x_{1;4} \end{matrix} \quad N \quad \begin{matrix} & x_{2;1} & x_{2;2} & x_{2;3} \end{matrix}$$
$$h \quad 0 \quad 0 \quad +1 \quad -1 \quad i \qquad h \quad +1 \quad 0 \quad -1 \quad i$$

$$= \begin{matrix} \\ x_{1;1} \\ x_{1;2} \\ x_{1;3} \\ x_{1;4} \end{matrix} \begin{matrix} \begin{matrix} x_{2;1} & x_{2;2} & x_{2;3} \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ +1 & 0 & -1 \\ -1 & 0 & +1 \end{bmatrix} \end{matrix}$$

Notice that the coecients in the matrix match the highlighted coecients in the dierence of di erences that was expressed a few sentences previously. The prior of this interaction contrast is shown in the bottom left histogram of Fig. 19.5. The mean of 34,600 indicates that the dierence between FT1 and FT2 is about 34,600 greater for CHEM than for THTR. The 95% HDI clearly excludes zero, indicating that this interaction contrast is credibly non-zero.

Interaction contrasts can involve "complex" comparisons just as simply as pairwise comparisons. For example, suppose we are interested in comparing BFIN against the

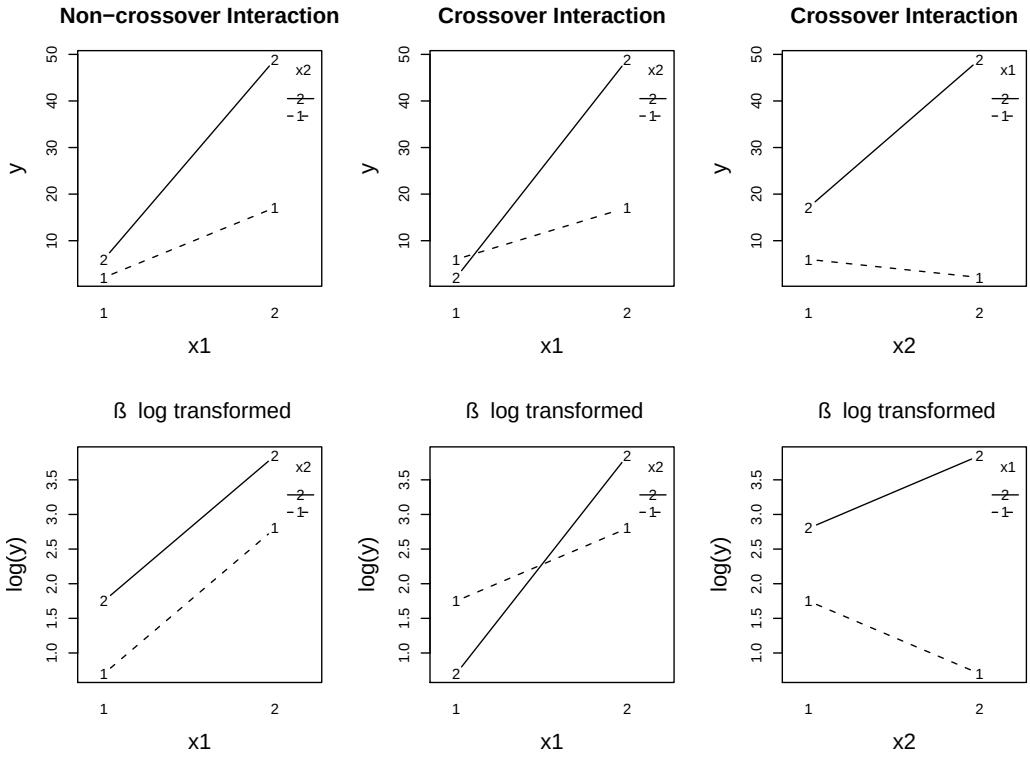

Figure 19.6: Top row shows means of original data; bottom row shows means of logarithmically transformed data. Left column shows a non-crossover interaction; middle and right columns show the same crossover interaction plotted against x1 or x2.

average of the other, non-business departments, specified a contrast between FT1 and lesser ranks. This interaction contrast is expressed as $(1/3, 1/3, 1/3, -1, -1, -1)$. The posterior of this contrast is shown in the bottom right histogram of Figure 19.5. (The label on the x-axis exceeds the margins of the figure because there are twelve combinations of levels involved in the contrast specification.) The result suggests that the larger difference, between FT1 and other ranks, in BFIN than in other departments has substantial uncertainty. Therefore we would not want to conclude that the interaction contrast is credibly non-zero. Exercise 19.2 gives you a hand-on practice with specification of interaction contrasts.

19.1.5 Non-crossover interactions, rescaling, and homogeneous variances

When interpreting interactions, it can be important to consider the scale on which the data are measured. This is because an interaction means non-additivity in the current scale. If the data are re-scaled by a non-linearly transformed to a different scale, then the non-additivity can also change.

Consider an example, using utterly fictional numbers merely for illustration. Suppose the average salary of Democratic women is 10 monetary units, for Democratic men it's 12 units, for Republican women it's 15 units, and for Republican men it's 18 units. These data indicate that there is a non-additive interaction of political party and gender, because the

change in pay from women to men is 2 units for Democrats, but 3 units for Republicans. Another way of describing the interaction is to notice that the change in pay from Democrats to Republicans is 5 units for women but 6 units for men. A reader might be tempted to interpret the interaction as indicating some extra advantage attained by Republican men, or some special disadvantage suffered by Democratic women. But such an interpretation may be inappropriate, because a mere rescaling of the data makes the interaction disappear, as will be described next.

Salary increases and comparisons are often measured by percentages and ratios, not by additive or subtractive differences. Consider the salary data in percentage terms. Among Democrats, men make 20% more than women. Among Republicans, men again make 20% more than the women. Among women, Republicans make 50% more than Democrats. Among men, Republicans again make 50% more than Democrats. Stated in ratio terms, there is no interaction of gender and political party: Change from female to male predicts a 20% increase in salary regardless of party, and change from Democrat to Republican predicts a 50% increase in salary regardless of gender.

Equal ratios are transformed to equal distances by a logarithmic transformation. If we measure salary in terms of the logarithm of monetary units, then the salary of Democratic women is $\log_{10}(10) = 1.000$, the salary of Democratic men is $\log_{10}(12) = 1.079$, the salary of Republican women is $\log_{10}(15) = 1.176$, and the salary of Republican men is $\log_{10}(18) = 1.255$. With this logarithmic scaling, the increase in salary of women to men is 0.079 for both parties, and the increase from Democrat to Republican in 0.176 for both genders. In other words, when salary is measured on a logarithmic scale, there is no interaction of gender and political party.

It may seem strange to measure salary on a logarithmic scale, but there are many situations for which the scale is arbitrary. The pitch of a sound can be measured in terms of frequency (i.e., cycles per second), or in terms of perceived pitch, which is essentially the logarithm of the frequency. The magnitude of an earthquake can be measured by its energy, or by its value on the Richter scale, which is the logarithm of energy. The pace of a dragster on a race track can be measured by the average speed during the run, or by the duration from start to finish (which is the reciprocal of average speed). Thus, measurement scales are not unique, and are instead determined by convention.

The general issue is illustrated in Figure 19.6. Suppose the predictor x_1 has two levels, as does predictor x_2. Suppose we have three data points at each combination of levels, yielding twelve data points altogether. The means at each combination of levels are shown in the top-left graph of Figure 19.6. You can see that there is an interaction, with the effect of x_1 being bigger when $x_2 = 2$ than when $x_2 = 1$. But this interaction goes away when the data are transformed by taking the logarithm, as shown in the lower left graph. Each individual data point was transformed, and then the means in each cell were computed. Of course, the transformation can go the other way: Data with no interaction, as in the lower-left plot, can be made to have an interaction when they are scaled as in the upper-left plot, via an exponential transformation.

The transformability from interaction to non-interaction is only possible for non-crossover interactions. This terminology is merely a description of the graph: The lines do not cross over each other (and they have the same sign slope). In this situation, the y-axis can have different portions stretched or shrunken so that the lines become parallel. If, however, the lines cross, as in the middle column of Figure 6.19, then there is no way to uncross the lines merely by stretching or shrinking intervals of the y-axis. The right column of Figure 19.6 shows the same data as the middle column, but plotted with the roles of x_1

and x_2 exchanged. When plotted this way, the lines do not cross, but they do have opposite-sign slopes (i.e., one slope is positive and the other slope is negative). There is no way that stretching or shrinking the y-axis can change the signs of the slopes, hence the interaction cannot be removed merely by transforming the data. Because these data have crossing lines when plotted as in the middle column, they are said to have a crossover interaction even when they are plotted as in the right column. (Test your understanding: Is the interaction in Figure 19.1 a crossover interaction?)

It is important to note that the transformation applies to individual raw data values, not to the means of the conditions. A consequence of transforming the data, therefore, is changes in the variances of the data within each condition. For example, suppose one condition has data values of 100, 110, and 120, while a second condition has data values of 1100, 1110, and 1120. For both conditions, the variance is 66.7, i.e., there is homogeneity of variance. When the data are logarithmically transformed, the variance of the first group becomes $1.05e{-3}$, but the variance of the second group becomes two orders of magnitude smaller, namely $1.02e{-5}$; i.e., there is not homogeneity of variance.

Therefore, when applying the hierarchical model of Figure 19.1, we must be aware that it assumes homogeneity of variance. If we transform the data, we are changing the variances within the levels of the predictors. The transformed variances might or might not be fairly homogeneous. If they are not, then either the data should be transformed in such a way as to respect homogeneity of variance, or the model should be changed to allow unequal variances.

The models we have been using also assume a normal likelihood function, which means that the data at any level of the predictors should be normally distributed. When the data are transformed to a different scale, the shape of their distribution also changes. If the distributions become radically non-normal, it may be misleading to use a model with a normal likelihood function. For a discussion of these issues, review Section 15.1.4, p. 326.

In summary, this section has made two main points. First, if you have a non-crossover interaction, be careful what you claim about it. A non-crossover interaction merely means non-additivity in the scale you are using. If this scale is the only meaningful scale, or if the this scale is the overwhelmingly dominant scale used in the field of research, then you can cautiously interpret the non-additive interaction with respect to that scale. But if transformed scales are reasonable, then keep in mind that non-additivity is scale-specific, and there might be no interaction in a different scale. With a crossover interaction, however, no rescaling can undo the interaction. Second, non-linear transformations change the within-cell variances and the shapes of the within-cell distributions. Be sure that the model you are using is appropriate to the homogeneity or non-homogeneity of variances in the data, and to the shapes of the distributions, on whatever scale you are using. Exercise 19.1 has you consider these issues “hands on”.

19.2 Repeated measures, a.k.a. within-subject designs

In many situations, a single “subject” contributes data to multiple levels of the predictors. For example, suppose we are interested in how quickly people can press a button in response to a stimulus onset. The stimulus could appear in the visual modality as a light, or in the auditory modality as a tone. The subject could respond with his/her dominant hand, or with his/her non-dominant hand. Thus, there are two nominal predictors, namely modality and hand. The new aspect is that a single subject contributes data to all combinations

of the predictors. On many successive trials, the subject gets either a tone or light, and is instructed to respond with either the dominant or non-dominant hand. Because every subject is measured many times, this situation is sometimes called a "repeated measures" design. Because the levels of the predictors change within subjects, this situation is also called a "within subject" design. I favor the latter terminology because it more explicitly connotes the essential aspect of the design, that the same subject contributes data in more than one condition. Within-subject designs are contrasted with "between-subject" designs, in which different subjects contribute data to different levels of the predictors.

When every subject contributes many data points to every combination of predictors, then the model of the situation is a straightforward extension of the models we've already considered. We merely add "subject" as another predictor in the model, with each individual subject being a level of the predictor. If there is one predictor other than subject, the model becomes

$$y = \beta_0 + \beta_1 x_1 + \beta_S x_S + \beta_{1S} x_{1S}$$

This is exactly the two-predictor model we have already considered, with the second predictor being subject. When there are two predictors other than subject, the model becomes

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_S x_S + \beta_{12} x_{12} + \beta_{1S} x_{1S} + \beta_{2S} x_{2S} + \beta_{12S} x_{12S}$$

This model includes all the two-way interactions of the factors plus the three-way interaction. Again, subject merely plays the role of the third predictor.

The model above, that includes all the high-order interactions with subject, is fine in principle but may be overkill in practice. Unless you have specific theoretical motivations to seek out and interpret high-order interactions of subject with other predictors, there is little reason to model them, and difficulty making sense of them even if you did model them. Instead, if you have many data points from each subject in every cell, an alternative approach is to apply a Bayesian ANOVA model to each subject's data, and then put a higher-order prior across the subject parameter estimates, so that different subjects mutually inform each other's estimates and provide a stable group-level estimate. Thus, every subject has a baseline, β_{0s}, and there is a higher-order, group-level prior on the distribution of β_{0s} across subjects. Each predictor also has subject-specific deviations, with the effect of the j^{th} level of predictor 1 denoted $\beta_{1s[j]}$. Each of these effect parameters has a higher, group-level prior across subjects. (This was the modeling approach taken for repeated measures in simple linear regression in Section 16.3, p. 354.) Finally, the group-level effects have a hyperprior that provides shrinkage on the effects of a predictor. In other words, the shrinkage prior, on effects of a predictor, is set at the group level, not at the subject level.

There are other situations, however, in which each subject contributes only one datum to a combination of the other predictors. For example, in the case of the response-time study described above, perhaps we have only the median response time of the subject in each combination of hand and modality. As another example, suppose the value to be predicted is IQ, as measured by a lengthy exam, with one predictor being noisy versus quiet exam environment, and the other predictor being paper versus computerized exam format. Although it is conceivable that subjects could be repeatedly tested in each condition, it would be challenging enough to get people to sit through all the combinations even once. Thus, each subject would contribute one value to each condition.

In the situation when each subject contributes only one datum per condition, the models described above, with all the interaction terms, are not identifiable", meaning that there are

more parameters than data points. The simplest case of this is trying to estimate the mean and variance of a normal distribution from a single data point. A Bayesian analysis can still be conducted, but there will be high uncertainty in the parameter estimates, governed largely by the priors. Therefore, instead of attempting to estimate all the interactions of subjects with other predictors, we assume a simple model in which the only influence of subjects is on the baseline:

$$y = \beta_0 + \beta_S x_S + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_{12}$$

In other words, we assume a main effect of subject, but no interaction of subject with other predictors. In this model, the subject effect (deflection) is constant across treatments, and the treatment effects (deflections) are constant across subjects. Notice that the model makes no requirement that every subject contributes a datum to every condition. Indeed, the model allows zero or more than one datum per subject per condition. As mentioned earlier, the computations in Bayesian ANOVA make no assumptions or requirements that the design is "balanced". If you do have many observations per subject in every combination of predictors, then one of the previously described models may be considered.

19.2.1 Why use a within-subject design? And why not?

The primary reason to use a within-subject design is that you can achieve much greater precision in the estimates of the effects than in a between-subject design. For example, suppose you are interested in measuring the effect on response time of using the dominant versus non-dominant hand. Suppose there is a population of subjects from whom you could measure data. If we could measure every subject in every condition, we would know that for the first subject, his or her response times for dominant and non-dominant hands are 300 and 320 msec. For the second subject, the response times are 350 and 370. For the third subject, the response times are 400 and 420, and for the fourth subject, the response times are 450 and 470. Thus, for every subject, the difference between dominant and non-dominant hands is 20 msec. Suppose we have the resources to measure only two data points in each condition. We measure response times from the dominant hands of two subjects. Should we measure response times from the non-dominant hands of the same two subjects, or the non-dominant hands of two other subjects? If we measure from the same two subjects, then the estimated effect for each subject is 20 msec, and we have high certainty in the magnitude of the effect. If we measure from two other subjects, then the estimated effect of dominant versus non-dominant hand is the average of the first two subjects versus the average of the second two subjects, and the difference is badly affected by random sampling. The between-subject design yields low precision in the estimate of the effect. Exercise 19.3 has you examine, hands on, a case of this improvement in precision.

Because of the gain in precision, it is desirable to use within-subject designs. But there are many dangers of within-subject designs that need to be considered before they are applied in any particular situation. The key problem is that, in most situations, when you measure the subject you change the subject, and therefore subsequent measurements are not measuring the same subject. The simplest examples of this are mere fatigue or generic practice effects. In measures of response time, if you measure repeatedly from the same subject, you will find improvement over the first several trials because of the subject gaining practice with the task, but after a while, the subject tires, there will be a

decline in performance. The problem is that if you measure the dominant hand in the early trials, and the nondominant hand in the later trials, then the effect of practice or fatigue will contaminate the effect of handedness. The repeated measurement process adds a contaminates the measure that is supposed to be a significant predictor.

Practice and fatigue effects can be overcome by randomly distributing and repeating the conditions throughout the repeated measures, so that the practice and fatigue effects influence all conditions equally. Thus, if practice improves both the dominant and nondominant hand by 50 msec, then the difference between dominant and nondominant hands is unaffected by practice. But practice might affect the nondominant hand much more than the dominant hand. You can imagine that in complex designs with many predictors, each with many levels, in can become difficult to justify an assumption that repeated measures have comparable effects on all conditions.

Worse yet, in some situations there can be differential carryover effects from one condition to the next. For example, having just experienced practice in the visual modality with the nondominant hand might improve subsequent performance in the auditory modality with the nondominant hand, but might not improve subsequent performance in the visual modality with the dominant hand. Thus, the carryover effect is different for different subsequent conditions.

When you suspect strong differential carryover effects, you may be able to explicitly manipulate the ordering of the conditions and measure the carryover effects, but this might be impossible mathematically and impractical, depending on the specifics of your situation. In this case, you must revert to a between subjects design and simply include many subjects to attenuate between-subject noise.

In general, all the models we have been using assume independence of observations. The probability of the collection of data is the product of the probabilities of the individual data points. When we use repeated measures, this assumption is much less easy to justify. On the one hand, when we repeatedly flip a coin, we might be safe to assume that its underlying bias does not change much from one flip to the next. But, on the other hand, when we repeatedly test the response time of a human subject, it is less easy to justify an assumption that the underlying response time remains unaffected by the previous trial. Researchers will often make the assumption of independence merely as an approximation of convenience, hoping that by arranging conditions randomly across many repeated measures, the differential carryover effects will be minimized.

19.3 R code

19.3.1 Bayesian two-factor ANOVA

Several implementation details of the program are the same as the oneway ANOVA program of the previous chapter:

Data are normalized so that prior constants can be more generic.

Initialization of chains is based on the data. It is important to do this, otherwise burn-in can take forever.

Because there is nasty autocorrelation, we use a large thinning constant and we also use multiple chains. For a reminder of the issues of burn-in and thinning, see Section 23.2, p. 510.

A new detail arises in how the uncentered parameter estimates are re-centered to respect the sum-to-zero constraints. The uncentered estimates in BUGS are a0, a1[], a2[], and a1a2[,]. By definition of the ANOVA model, the predicted mean of cell j is

```
m[i,j] = a0 + a1[i] + a2[j] + a1a2[i,j]
```

We use these predicted means to construct the zero-centered parameters. First, b0 is the mean across all the predicted means:

```
b0 = mean( m[,] )
```

Then the main effect deflections are the marginal means minus the overall mean:

```
b1[i] = mean( m[i,] ) - b0
b2[j] = mean( m[,j] ) - b0
```

It is easy (honest!) to check that the those deflections do indeed sum to zero; i.e., sum(b1[]) = 0 and sum(b2[]) = 0. Finally, the interaction deflections are the residuals after the additive effect of b1 and b2 is taken into account:

```
b1b2[i,j] = m[i,j] - ( b0 + b1[i] + b2[j] )
```

Again, it is easy to check that the rows and columns of b1b2[,] all sum to zero.

In the data section of the program, one option is to load data from the article of Qian and Shen (2007). The program here uses a hierarchical structure similar to that used by Qian and Shen (2007), but their program did not re-center the parameters as is done here. It may be instructive to compare the results of the program with the results reported by Qian and Shen (2007).

BUGS for many factors. The program below applies only for cases of two nominal predictors. If you have many nominal predictors, along with their two-way, three-way, and higher-order interactions, it becomes unwieldy to index and separately name all the deflection parameters. Instead, it can be more elegant to use a technique of "dummy coding", whereby we essentially revert back to using vectors for coding the values of the predictors instead of integer indices. That is, if x_1 = level 2 is coded by the "dummy" vector $\langle 0, 1, 0, \ldots \rangle$ instead of by the integer index 2. Interactions are represented by matrices of dummy codes that have been flattened into vectors. For an example of programming this technique in BUGS, see Ntzoufras (2009, Ch. 6). Unfortunately, those examples do not incorporate the higher-level prior structure emphasized in Figure 19.2.

(ANOVAtwowayBRugs.R)

```
graphics.off()
rm(list=ls(all=TRUE))
fnroot = "ANOVAtwowayBrugs"
library(BRugs)              # Kruschke, J. K. (2010). Doing Bayesian data analysis:
                              # A Tutorial with R and BUGS. Academic Press / Elsevier.
#------------------------------------------------------------------------------
# THE MODEL.

modelstring = "
# BUGS model specification begins here...
model {
  for ( i in 1:Ntotal ) {
    y[i] ~ dnorm( mu[i] , tau )
    mu[i] <- a0 + a1[x1[i]] + a2[x2[i]] + a1a2[x1[i],x2[i]]
```

```
    }
    #
    tau <- pow( sigma , -2 )
    sigma ~ dunif(0,10) # y values are assumed to be standardized
    #
    a0 ~ dnorm(0,0.001) # y values are assumed to be standardized
    #
    for ( j1 in 1:Nx1Lvl ) { a1[j1] ~ dnorm( 0.0 , a1tau ) }
    a1tau <- 1 / pow( a1SD , 2 )
    a1SD <- abs( a1SDunabs ) + .1
    a1SDunabs ~ dt( 0 , 0.001 , 2 )
    #
    for ( j2 in 1:Nx2Lvl ) { a2[j2] ~ dnorm( 0.0 , a2tau ) }
    a2tau <- 1 / pow( a2SD , 2 )
    a2SD <- abs( a2SDunabs ) + .1
    a2SDunabs ~ dt( 0 , 0.001 , 2 )
    #
    for ( j1 in 1:Nx1Lvl ) { for ( j2 in 1:Nx2Lvl ) {
      a1a2[j1,j2] ~ dnorm( 0.0 , a1a2tau )
    } }
    a1a2tau <- 1 / pow( a1a2SD , 2 )
    a1a2SD <- abs( a1a2SDunabs ) + .1
    a1a2SDunabs ~ dt( 0 , 0.001 , 2 )
  }
  # ... end BUGS model specification
  " # close quote for modelstring
  # Write model to a file, and send to BUGS:
  writeLines(modelstring,con="model.txt")
  modelCheck( "model.txt" )

  #------------------------------------------------- ----------------------------
  # THE DATA.
  # Specify data source:
  dataSource = c( "QianS2007" , "Salary" , "Random" , "Ex19.3"       )[4]

  # Load the data:
  if ( dataSource == "QianS2007" ) {
    fnroot = paste( fnroot , dataSource , sep="" )
    datarecord = read.table( "QianS2007SeaweedData.txt" , he     ader=TRUE , sep="," )
    # Logistic transform the COVER value:
    # Used by Appendix 3 of QianS2007 to replicate Ramsey and Schafer (2002).
    datarecord$COVER = -log( ( 100 / datarecord$COVER ) - 1 )
    y = as.numeric(datarecord$COVER)
    x1 = as.numeric(datarecord$TREAT)
    x1names = levels(datarecord$TREAT)
    x2 = as.numeric(datarecord$BLOCK)
    x2names = levels(datarecord$BLOCK)
    Ntotal = length(y)
    Nx1Lvl = length(unique(x1))
    Nx2Lvl = length(unique(x2))
    x1contrastList = list( f_Effect=c( 1/2 , -1/2 , 0 , 1/2 , -1/2 ,              0 ) ,
                           F_Effect=c( 0 , 1/2 , -1/2 , 0 , 1/2 , -1/2 ) ,
                           L_Effect=c( 1/3 , 1/3 , 1/3 , -1/3 , -1/3 , -1/3 ) )
    x2contrastList = NULL # list( vector(length=Nx2Lvl) )
    x1x2contrastList = NULL # list( matrix( 1:(Nx1Lvl*Nx2Lvl)          , nrow=Nx1Lvl ) )
  }

  if ( dataSource == "Salary" ) {
    fnroot = paste( fnroot , dataSource , sep="" )
```

```
  datarecord = read.table( "Salary.csv" , header=TRUE , sep="," )
  y = as.numeric(datarecord$Salary)
  if ( F ) { # take log10 of salary
    y = log10( y )
    fnroot = paste( fnroot , "Log10" , sep="" )
  }
  x1 = as.numeric(datarecord$Org)
  x1names = levels(datarecord$Org)
  x2 = as.numeric(datarecord$Post)
  x2names = levels(datarecord$Post)
  Ntotal = length(y)
  Nx1Lvl = length(unique(x1))
  Nx2Lvl = length(unique(x2))
  x1contrastList = list( BFINvCEDP = c( 1 , -1 , 0 , 0 ) ,
                         CEDPvTHTR = c( 0 , 1 , 0 , -1 ) )
  x2contrastList = list( FT1vFT2 = c( 1 , -1 , 0 ) , FT2vFT3 = c(0,1 ,-1) )
  x1x2contrastList = list(
      CHEMvTHTRxFT1vFT3 = outer( c(0,0,+1,-1) , c(+1,0,-1) ) ,
      BFINvOTHxFT1vOTH = outer( c(+1,-1/3,-1/3,-1/3) , c(+1,-1 /2,-1/2) ) )
}

if ( dataSource == "Random" ) {
  fnroot = paste( fnroot , dataSource , sep="" )
  set.seed(47405)
  ysdtrue = 3.0
  a0true = 100
  a1true = c( 2 , 0 , -2 ) # sum to zero
  a2true = c( 3 , 1 , -1 , -3 ) # sum to zero
  a1a2true = matrix( c( 1,-1,0, -1,1,0, 0,0,0, 0,0,0 ),# row an d col sum to zero
                     nrow=length(a1true) , ncol=length(a2true) , byrow=F )
  npercell = 8
  datarecord = matrix( 0, ncol=3 , nrow=length(a1true)*leng th(a2true)*npercell )
  colnames(datarecord) = c("y","x1","x2")
  rowidx = 0
  for ( x1idx in 1:length(a1true) ) {
    for ( x2idx in 1:length(a2true) ) {
      for ( subjidx in 1:npercell ) {
        rowidx = rowidx + 1
        datarecord[rowidx,"x1"] = x1idx
        datarecord[rowidx,"x2"] = x2idx
        datarecord[rowidx,"y"] = ( a0true + a1true[x1idx] + a2true [x2idx]
                                   + a1a2true[x1idx,x2idx] + rnorm(1,0,ysdtrue) )
      }
    }
  }
  datarecord = data.frame( y=datarecord[,"y"] ,
                           x1=as.factor(datarecord[,"x1"]) ,
                           x2=as.factor(datarecord[,"x2"]) )
  y = as.numeric(datarecord$y)
  x1 = as.numeric(datarecord$x1)
  x1names = levels(datarecord$x1)
  x2 = as.numeric(datarecord$x2)
  x2names = levels(datarecord$x2)
  Ntotal = length(y)
  Nx1Lvl = length(unique(x1))
  Nx2Lvl = length(unique(x2))
  x1contrastList = list( X1_1v3 = c( 1 , 0 , -1 ) ) #
  x2contrastList =  list( X2_12v34 = c( 1/2 , 1/2 , -1/2 , -1/2 ) ) #
  x1x2contrastList = list(
```

```
    IC_11v22 = outer( c(1,-1,0) , c(1,-1,0,0) ) ,
    IC_23v34 = outer( c(0,1,-1) , c(0,0,1,-1) )
  )
}

# Load the data:
if ( dataSource == "Ex19.3" ) {
  fnroot = paste( fnroot , dataSource , sep="" )
  y = c( 101,102,103,105,104, 104,105,107,106,108, 105,107 ,106,108,109, 109,108,110,111,112 )
  x1 = c( 1,1,1,1,1, 1,1,1,1,1, 2,2,2,2,2, 2,2,2,2,2 )
  x2 = c( 1,1,1,1,1, 2,2,2,2,2, 1,1,1,1,1, 2,2,2,2,2 )
  # S = c( 1,2,3,4,5, 1,2,3,4,5, 1,2,3,4,5, 1,2,3,4,5 )
  x1names = c("x1.1","x1.2")
  x2names = c("x2.1","x2.2")
  # Snames = c("S1","S2","S3","S4","S5")
  Ntotal = length(y)
  Nx1Lvl = length(unique(x1))
  Nx2Lvl = length(unique(x2))
  # NSLvl = length(unique(S))
  x1contrastList = list( X1.2vX1.1 = c( -1 , 1 ) )
  x2contrastList = list( X2.2vX2.1 = c( -1 , 1 ) )
  x1x2contrastList = NULL # list( matrix( 1:(Nx1Lvl*Nx2Lvl) , nrow=Nx1Lvl ) )
}

# Specify the data in a form that is compatible with BRugs model, as a list:
ySDorig = sd(y)
yMorig = mean(y)
z = ( y - yMorig ) / ySDorig
datalist = list(
  y = z ,
  x1 = x1 ,
  x2 = x2 ,
  Ntotal = Ntotal ,
  Nx1Lvl = Nx1Lvl ,
  Nx2Lvl = Nx2Lvl
)
# Get the data into BRugs:
modelData( bugsData( datalist ) )

#------------------------------------------------ ----------
# INTIALIZE THE CHAINS.

# Autocorrelation within chains is large, so use several chains to reduce
# degree of thinning. But we still have to burn-in all the chains, which takes
# more time with more chains.
nchain = 10
modelCompile( numChains = nchain )

if ( F ) {
  modelGenInits() # often won't work for diffuse prior
} else {
  #  initialization based on data
  theData = data.frame( y=datalist$y , x1=factor(x1,labels=x1names) ,
                        x2=factor(x2,labels=x2names) )
  a0 = mean( theData$y )
  a1 = aggregate( theData$y , list( theData$x1 ) , mean )[,2] - a0
  a2 = aggregate( theData$y , list( theData$x2 ) , mean )[,2] - a0
  linpred = as.vector( outer( a1 , a2 , "+" ) + a0 )
  a1a2 = aggregate( theData$y, list(theData$x1,theData$x2), mean)[,3] - linpred
```

```
  genInitList <- function() {
    return(
      list(
        a0 = a0 ,
        a1 = a1 ,
        a2 = a2 ,
        a1a2 = matrix( a1a2 , nrow=Nx1Lvl , ncol=Nx2Lvl ) ,
        sigma = sd(theData$y)/2 , # lazy
        a1SDunabs = sd(a1) ,
        a2SDunabs = sd(a2) ,
        a1a2SDunabs = sd(a1a2)
      )
    )
  }
  for ( chainIdx in 1 : nchain ) {
    modelInits( bugsInits( genInitList ) )
  }
}

#-------------------------------------------------                    ----------------------------
# RUN THE CHAINS

# burn in
BurnInSteps = 10000
modelUpdate( BurnInSteps )
# actual samples
samplesSet( c( "a0" , "a1" , "a2" , "a1a2" ,
                "sigma" , "a1SD" , "a2SD" , "a1a2SD" ) )
stepsPerChain = ceiling(2000/nchain)
thinStep = 500 # 750
modelUpdate( stepsPerChain , thin=thinStep )

#-------------------------------------------------                    ----------------------------
# EXAMINE THE RESULTS

source("plotChains.R")
source("plotPost.R")

checkConvergence = F
if ( checkConvergence ) {
  sumInfo = plotChains( "a0" , saveplots=F , filenameroot=fn      root )
  sumInfo = plotChains( "a1" , saveplots=F , filenameroot=fn      root )
  sumInfo = plotChains( "a2" , saveplots=F , filenameroot=fn      root )
  sumInfo = plotChains( "a1a2" , saveplots=F , filenameroot=      fnroot )
  readline("Press any key to clear graphics and continue")
  graphics.off()
  sumInfo = plotChains( "sigma" , saveplots=F , filenameroot      =fnroot )
  sumInfo = plotChains( "a1SD" , saveplots=F , filenameroot=      fnroot )
  sumInfo = plotChains( "a2SD" , saveplots=F , filenameroot=      fnroot )
  sumInfo = plotChains( "a1a2SD" , saveplots=F , filenameroo      t=fnroot )
  readline("Press any key to clear graphics and continue")
  graphics.off()
}

# Extract and plot the SDs:
sigmaSample = samplesSample("sigma")
a1SDSample = samplesSample("a1SD")
a2SDSample = samplesSample("a2SD")
a1a2SDSample = samplesSample("a1a2SD")
```

```
windows()
layout( matrix(1:4,nrow=2) )
par( mar=c(3,1,2.5,0) , mgp=c(2,0.7,0) )
plotPost( sigmaSample , xlab="sigma" , main="Cell SD" , bre aks=30 )
plotPost( a1SDSample , xlab="a1SD" , main="a1 SD" , breaks= 30 )
plotPost( a2SDSample , xlab="a2SD" , main="a2 SD" , breaks= 30 )
plotPost( a1a2SDSample , xlab="a1a2SD" , main="Interacti on SD" , breaks=30 )
dev.copy2eps(file=paste(fnroot,"SD.eps",sep=""))

# Extract a values:
a0Sample = samplesSample( "a0" )
chainLength = length(a0Sample)
a1Sample = array( 0 , dim=c( datalist$Nx1Lvl , chainLength ) )
for ( x1idx in 1:datalist$Nx1Lvl ) {
  a1Sample[x1idx,] = samplesSample( paste("a1[",x1idx,"] ",sep="") )
}
a2Sample = array( 0 , dim=c( datalist$Nx2Lvl , chainLength ) )
for ( x2idx in 1:datalist$Nx2Lvl ) {
  a2Sample[x2idx,] = samplesSample( paste("a2[",x2idx,"] ",sep="") )
}
a1a2Sample = array(0, dim=c( datalist$Nx1Lvl , datalist$N x2Lvl , chainLength ) )
for ( x1idx in 1:datalist$Nx1Lvl ) {
  for ( x2idx in 1:datalist$Nx2Lvl ) {
    a1a2Sample[x1idx,x2idx,] = samplesSample( paste( "a1a2[ ",x1idx,",",x2idx,"]",
                                                     sep="" ) )
  }
}

# Convert to zero-centered b values:
m12Sample = array( 0, dim=c( datalist$Nx1Lvl , datalist$Nx 2Lvl , chainLength ) )
for ( stepIdx in 1:chainLength ) {
  m12Sample[,,stepIdx ] = ( a0Sample[stepIdx]
                            + outer( a1Sample[,stepIdx] ,
                                     a2Sample[,stepIdx] , "+" )
                            + a1a2Sample[,,stepIdx] )
}
b0Sample = apply( m12Sample , 3 , mean )
b1Sample = ( apply( m12Sample , c(1,3) , mean )
             - matrix(rep( b0Sample ,Nx1Lvl),nrow=Nx1Lvl,byrow=T) )
b2Sample = ( apply( m12Sample , c(2,3) , mean )
             - matrix(rep( b0Sample ,Nx2Lvl),nrow=Nx2Lvl,byrow=T) )
linpredSample = array(0,dim=c(datalist$Nx1Lvl,datalis t$Nx2Lvl,chainLength))
for ( stepIdx in 1:chainLength ) {
  linpredSample[,,stepIdx ] = ( b0Sample[stepIdx]
                                + outer( b1Sample[,stepIdx] ,
                                         b2Sample[,stepIdx] , "+" ) )
}
b1b2Sample = m12Sample - linpredSample
# Convert from standardized b values to original scale b valu es:
b0Sample = b0Sample * ySDorig + yMorig
b1Sample = b1Sample * ySDorig
b2Sample = b2Sample * ySDorig
b1b2Sample = b1b2Sample * ySDorig

# Plot b values:
windows((datalist$Nx1Lvl+1)*2.75,(datalist$Nx2Lvl+1 )*2.0)
layoutMat = matrix( 0 , nrow=(datalist$Nx2Lvl+1) , ncol=(d atalist$Nx1Lvl+1) )
layoutMat[1,1] = 1
layoutMat[1,2:(datalist$Nx1Lvl+1)] = 1:datalist$Nx1Lv l + 1
```

```
layoutMat[2:(datalist$Nx2Lvl+1),1] = 1:datalist$Nx2Lv l + (datalist$Nx1Lvl + 1)
layoutMat[2:(datalist$Nx2Lvl+1),2:(datalist$Nx1Lvl+ 1)] = matrix(
    1:(datalist$Nx1Lvl*datalist$Nx2Lvl) + (datalist$Nx2Lv l+datalist$Nx1Lvl+1) ,
    ncol=datalist$Nx1Lvl , byrow=T )
layout( layoutMat )
par( mar=c(4,0.5,2.5,0.5) , mgp=c(2,0.7,0) )
histinfo = plotPost( b0Sample , xlab=expression(beta * 0) , main="Baseline" ,
                     breaks=30 )
for ( x1idx in 1:datalist$Nx1Lvl ) {
  histinfo = plotPost( b1Sample[x1idx,] , xlab=bquote(beta *1[.(x1idx)]) ,
                       main=paste("x1:",x1names[x1idx]) , breaks=30 )
}
for ( x2idx in 1:datalist$Nx2Lvl ) {
  histinfo = plotPost( b2Sample[x2idx,] , xlab=bquote(beta *2[.(x2idx)]) ,
                       main=paste("x2:",x2names[x2idx]) , breaks=30 )
}
for ( x2idx in 1:datalist$Nx2Lvl ) {
  for ( x1idx in 1:datalist$Nx1Lvl ) {
    histinfo = plotPost( b1b2Sample[x1idx,x2idx,] , breaks=3 0 ,
                xlab=bquote(beta*12[.(x1idx)*","*.(x2idx)]) ,
                main=paste("x1:",x1names[x1idx],", x2:",x2names[x2id x]) )
  }
}
dev.copy2eps(file=paste(fnroot,"b.eps",sep=""))

# Display contrast analyses
nContrasts = length( x1contrastList )
if ( nContrasts > 0 ) {
  nPlotPerRow = 5
  nPlotRow = ceiling(nContrasts/nPlotPerRow)
  nPlotCol = ceiling(nContrasts/nPlotRow)
  windows(3.75*nPlotCol,2.5*nPlotRow)
  layout( matrix(1:(nPlotRow*nPlotCol),nrow=nPlotRow,n col=nPlotCol,byrow=T) )
  par( mar=c(4,0.5,2.5,0.5) , mgp=c(2,0.7,0) )
  for ( cIdx in 1:nContrasts ) {
    contrast = matrix( x1contrastList[[cIdx]],nrow=1) # make it a row matrix
    incIdx = contrast!=0
    histInfo = plotPost( contrast %*% b1Sample , compVal=0 , bre aks=30 ,
                xlab=paste( round(contrast[incIdx],2) , x1names[incIdx ] ,
                            c(rep("+",sum(incIdx)-1),"") , collapse=" " ) ,
                cex.lab = 1.0 ,
                main=paste( "X1 Contrast:", names(x1contrastList)[cIdx ] ) )
  }
  dev.copy2eps(file=paste(fnroot,"x1Contrasts.eps",se p=""))
}
#
nContrasts = length( x2contrastList )
if ( nContrasts > 0 ) {
  nPlotPerRow = 5
  nPlotRow = ceiling(nContrasts/nPlotPerRow)
  nPlotCol = ceiling(nContrasts/nPlotRow)
  windows(3.75*nPlotCol,2.5*nPlotRow)
  layout( matrix(1:(nPlotRow*nPlotCol),nrow=nPlotRow,n col=nPlotCol,byrow=T) )
  par( mar=c(4,0.5,2.5,0.5) , mgp=c(2,0.7,0) )
  for ( cIdx in 1:nContrasts ) {
    contrast = matrix( x2contrastList[[cIdx]],nrow=1) # make it a row matrix
    incIdx = contrast!=0
    histInfo = plotPost( contrast %*% b2Sample , compVal=0 , bre aks=30 ,
                xlab=paste( round(contrast[incIdx],2) , x2names[incIdx ] ,
```

```
                                    c(rep("+",sum(incIdx)-1),"") , collapse=" " ) ,
                    cex.lab = 1.0 ,
                    main=paste( "X2 Contrast:", names(x2contrastList)[cIdx ] ) )
    }
    dev.copy2eps(file=paste(fnroot,"x2Contrasts.eps",se p=""))
}
#
nContrasts = length( x1x2contrastList )
if ( nContrasts > 0 ) {
    nPlotPerRow = 5
    nPlotRow = ceiling(nContrasts/nPlotPerRow)
    nPlotCol = ceiling(nContrasts/nPlotRow)
    windows(3.75*nPlotCol,2.5*nPlotRow)
    layout( matrix(1:(nPlotRow*nPlotCol),nrow=nPlotRow,n col=nPlotCol,byrow=T) )
    par( mar=c(4,0.5,2.5,0.5) , mgp=c(2,0.7,0) )
    for ( cIdx in 1:nContrasts ) {
        contrast = x1x2contrastList[[cIdx]]
        contrastArr = array( rep(contrast,chainLength) ,
                                    dim=c(NROW(contrast),NCOL(contrast),chainLength) )
        contrastLab = ""
        for ( x1idx in 1:Nx1Lvl ) {
          for ( x2idx in 1:Nx2Lvl ) {
            if ( contrast[x1idx,x2idx] != 0 ) {
              contrastLab = paste( contrastLab , "+" ,
                                    signif(contrast[x1idx,x2idx],2) ,
                                    x1names[x1idx] , x2names[x2idx] )
            }
          }
        }
        histInfo = plotPost( apply( contrastArr * b1b2Sample , 3 , su m ) ,
                    compVal=0 , breaks=30 , xlab=contrastLab , cex.lab = 0.75 ,
                    main=paste( names(x1x2contrastList)[cIdx] ) )
    }
    dev.copy2eps(file=paste(fnroot,"x1x2Contrasts.eps", sep=""))
}

#=============================================================================
# Do NHST ANOVA:

theData = data.frame( y=y , x1=factor(x1,labels=x1names) ,
                                    x2=factor(x2,labels=x2names) )
windows()
interaction.plot( theData$x1 , theData$x2 , theData$y , ty pe="b" )
dev.copy2eps(file=paste(fnroot,"DataPlot.eps",sep=" "))
aovresult = aov( y ~ x1 * x2 , data = theData )
cat("\n------------------------------------------- ----------\n\n")
print( summary( aovresult ) )
cat("\n------------------------------------------- ----------\n\n")
print( model.tables( aovresult , type = "effects", se = TRUE ) , digits=3 )
cat("\n------------------------------------------- ----------\n\n")

#=============================================================================
```

19.4 Exercises

Exercise 19.1. [Purpose: Inspecting an interaction for transformed data.] Consider the data plotted in Figure 19.3, p. 426.

(A) Is the interaction a crossover interaction or not? Brie y explain your answer.

(B) Suppose we are interested in salaries thought of in terms of percentage (i.e., ratio) di erences rather than additive di erences. Therefore we take the logarithm, base 10, of the individual salaries (the R code has this option built into the data section, where the salary data are loaded). Run the analysis on the transformed data, producing the results and contrasts analogous to those in Figures 19.4 and 19.5. Do any of the conclusions change?

(C) Examine the within-cell variances in the original and in the transformed data. (Hint: Try using the aggregate function on the data. As a guide, see how the function is used to initialize a1a2. Instead of applying the mean to the aggregated data, apply the standard deviation. The result is the within-cell standard deviation. Are they all roughly the same?) Do the original or the transformed data better respect the assumption of homogeneous variances?

Exercise 19.2. [Purpose: Investigating a case of two factor Bayesian ANOVA.] In the data speci cation of the program in Section 19.3.4 (ANOVAtwowayBRugs.R) you can load data from Qian and Shen (2007), regarding how quickly seaweed regenerates in the presence of different types of grazers. Data were collected from eight distinct tidal areas of the Oregon coast; this predictor (x_2) is referred to as the "Block". In each of the eight Blocks, there were six di erent combinations of seaweed grazers established by the researchers. This predictor (x_1) had six levels: Control, with no grazers allowed; only small sh allowed; small and large sh allowed; only limpets allowed; limpets and small sh allowed; and, all three grazers allowed. The predicted variable was the percentage of the plot covered by regenerated seaweed, logarithmically transformed.

(A) Load the data and run the program. You will nd that there are too many levels of the two predictors to t all the posterior histograms into a single multi-panel display. Therefore, modify the plotPost.R program so that it produces only the mean and HDI limits, marked by a horizontal bar with a circle at the mean (without a histogram), and perhaps without a main title. Name your program something other than plotPost.R, and use it in the plotting section at the end of the program instead of plotPost. Show your results. (An secondary goal of this part of the exercise is to give you experience modifying graphics in R to suit your own purposes.) Hints: There are many ways to do this, but here are some options. To suppress plotting of the histogram, just set this argument in the hist function: plot=F . To suppress a title on a plot, just use the argument main="" . To adjust the font size, specify the "character expansion" cex for text, cex.lab for axis labels, and so forth. To reduce the margins around a plot, so there is more room for the plot itself, try variations of these margin speci cations: par(mar=c(2,0.5,1,0.5), mgp=c(0.5,0,0)) . The par command needs to be called before the plots are made.

(B) The program already includes contrasts that consider whether there is an e ect of small sh, and e ect of large sh, and an e ect of limpets. What conclusions do you reach from the posteriors of these contrasts?

(C) Construct a contrast of the average of Blocks 3 and 4 versus the average of Blocks 1 and 2. Show your speci cation, the graph of the posterior for the contrast, and state your conclusion.

(D) Is the e ect of limpets di erent in Block 6 than in Block 7? To answer this question,

construct an interaction contrast using an outer product. (Hint: Refer to the already-coded L_effect for the contrast that specifies the effect of limpets). Is the effect of small fish different in Blocks 1 and 7 than in Blocks 3 and 5? For both questions, show the contrasts vectors that you constructed and show the posterior of the contrast, and state your conclusion.

Exercise 19.3. [Purpose: Notice that within-subject designs can be more sensitive (hence more powerful) than between-subject designs.] Consider these data:

x_1	x_2	y	S
1	1	101	1
1	1	102	2
1	1	103	3
1	1	105	4
1	1	104	5
1	2	104	1
1	2	105	2
1	2	107	3
1	2	106	4
1	2	108	5
2	1	105	1
2	1	107	2
2	1	106	3
2	1	108	4
2	1	109	5
2	2	109	1
2	2	108	2
2	2	110	3
2	2	111	4
2	2	112	5

(A) Ignoring the last column, which indicates the subject who generated the data, conduct a Bayesian ANOVA using x_1 and x_2 as predictors of y. Show the code you used to load the data, and show the resulting posterior histograms of $_0$, $_{1;j}$, $_{2;k}$, and $_{1\ 2;jk}$. Also show the posterior of the contrast $_{1;2}$ $_{1;1}$ (i.e., the marginal difference between levels 1 and 2 of factor 1, also called the main effect of factor 1), and the posterior of the contrast $_{2;2}$ $_{2;1}$ (i.e., the marginal difference between levels 1 and 2 of factor 2, also called the main effect of factor 2).

(B) Now include the subject as a predictor, by expanding the model to include a deflection from baseline due to subject. (Do not include any subject interaction terms.) Again show the posteriors of the 's requested in the previous part. Are the certainties on the estimates and contrasts different than in the previous part? In what way, and why?

Hint regarding the answer: Figure 19.7 shows posterior histograms for the main effect of factor 2, when the data are considered to be between-subject or within-subject. Notice that the means are (essentially) the same in both histograms, but the uncertainties are very different!

Programming hints: The model specification without a subject factor is

```
mu[i] <- a0 + a1[x1[i]] + a2[x2[i]] + a1a2[x1[i],x2[i]]
```

but with a subject factor becomes

```
mu[i] <- a0 + a1[x1[i]] + a2[x2[i]] + a1a2[x1[i],x2[i]] + aS[          S[i]]
```

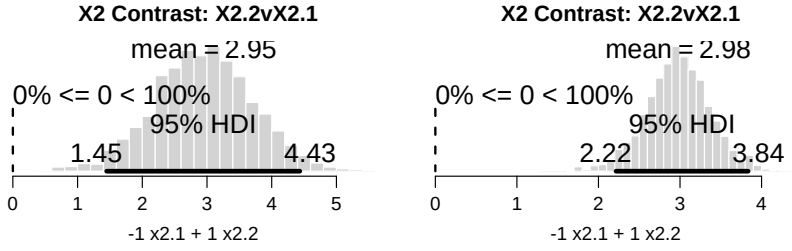

Figure 19.7: For Exercise 19.3. Left: Posterior for difference between levels of factor 2 when data are considered to be between-subject. Right: Posterior for difference between levels of factor 2 when data are considered to be within-subject. Notice that the means are (essentially) the same in both panels, but the uncertainties are very different!

where `S[i]` is the subject number for the i^{th} datum, and `aS[]` are the deflections from baseline for each subject. You must, of course, specify a prior on `aS[]` analogous to the prior on `a1[]`.

The conversion of the `a[]` values to zero-centered `b[]` values proceeds analogously to what was explained at the beginning of Section 19.3.1 (ANOVAtwowayBRugs.R). The code merely needs to be expanded to include the additional subject factor. Here is a guide: (ANOVAtwowayBRugsWithinSubj.R)

```
# Convert the a values to zero-centered b values.
# m12Sample is predicted cell means at every step in MCMC chain:
m12Sample = array( 0, dim=c( datalist$Nx1Lvl , datalist$Nx2Lvl ,
                             datalist$NSLvl , chainLength ) )
for ( stepIdx in 1:chainLength ) {
  for ( a1idx in 1:Nx1Lvl ) {
    for ( a2idx in 1:Nx2Lvl ) {
      for ( aSidx in 1:NSLvl ) {
        m12Sample[ a1idx , a2idx , aSidx , stepIdx ] = (
          a0Sample[stepIdx]
          + a1Sample[a1idx,stepIdx]
          + a2Sample[a2idx,stepIdx]
          + a1a2Sample[a1idx,a2idx,stepIdx]
          + aSSample[aSidx,stepIdx] )
      }
    }
  }
}

# b0Sample is mean of the cell means at every step in chain:
b0Sample = apply( m12Sample , 4 , mean )
# b1Sample is deflections of factor 1 marginal means from b0Sample:
b1Sample = ( apply( m12Sample , c(1,4) , mean )
             - matrix(rep( b0Sample ,Nx1Lvl),nrow=Nx1Lvl,byrow=T) )
# b2Sample is deflections of factor 2 marginal means from b0Sample:
b2Sample = ( apply( m12Sample , c(2,4) , mean )
             - matrix(rep( b0Sample ,Nx2Lvl),nrow=Nx2Lvl,byrow=T) )
# bSSample is deflections of factor S marginal means from b0Sample:
bSSample = ( apply( m12Sample , c(3,4) , mean )
             - matrix(rep( b0Sample ,NSLvl),nrow=NSLvl,byrow=T) )
# linpredSample is linear combination of the marginal effects:
linpredSample = 0*m12Sample
```

```
for ( stepIdx in 1:chainLength ) {
  for ( a1idx in 1:Nx1Lvl ) {
    for ( a2idx in 1:Nx2Lvl ) {
      for ( aSidx in 1:NSLvl ) {
        linpredSample[a1idx,a2idx,aSidx,stepIdx ] = (
            b0Sample[stepIdx]
            + b1Sample[a1idx,stepIdx]
            + b2Sample[a2idx,stepIdx]
            + bSSample[aSidx,stepIdx] )
      }
    }
  }
}
# b1b2Sample is the interaction deflection, i.e., the diffe          rence
# between the cell means and the linear combination:
b1b2Sample = apply( m12Sample - linpredSample , c(1,2,4) , m   ean )

# Convert from standardized b values to original scale b valu      es:
b0Sample = b0Sample * ySDorig + yMorig
b1Sample = b1Sample * ySDorig
b2Sample = b2Sample * ySDorig
bSSample = bSSample * ySDorig
b1b2Sample = b1b2Sample * ySDorig
```

Exercise 19.4. [Purpose: Power analysis for Bayesian ANOVA, for within-subject versus between-subject designs.] Conduct power analyses for the between-subject and within-subject versions of the previous exercise. Specifically, suppose that the goal is for the 95% HDI of the contrast on factor 2 to have a width of 2.0 or less. Conduct a prospective power analysis for this goal, for the within-subject version and the between-subject version. Caution: This exercise demands a lot of programming and could be time consuming, but the results drive home the point that within-subject designs can be more powerful than between-subject designs.

Chapter 20

Dichotomous Predicted Variable

Contents

> Fortune and Favor make fickle decrees, it's
> Heads or it's tails with no middle degrees.
> Flippant commandments decreed by law gods, have
> Reasons so rare they have minus log odds.

There are many situations in which the value to be predicted is dichotomous (instead of metric). For example, we might want to predict whether a patient is cured or not (a dichotomous outcome), on the basis of the dosage of drug administered and age of the patient (two metric predictors). What the model does, in this case, is generate a prediction of the probability that a patient is cured, given the specified dosage and age. In other situations, the predictors might be nominal. For example, we could predict cured versus not-cured on the basis of type of drug and gender of patient. The model is again the probability that a patient is cured, given the drug and the gender. We will denote such situations in this chapter.

The formal framework for this situation was presented in the second row of Table 14.1 on p. 312. As you may recall, the link function, which maps a linear combination of predictors to an outcome tendency, is a logistic function. Therefore the models we use in this chapter are referred to as cases of logistic regression or logistic ANOVA.

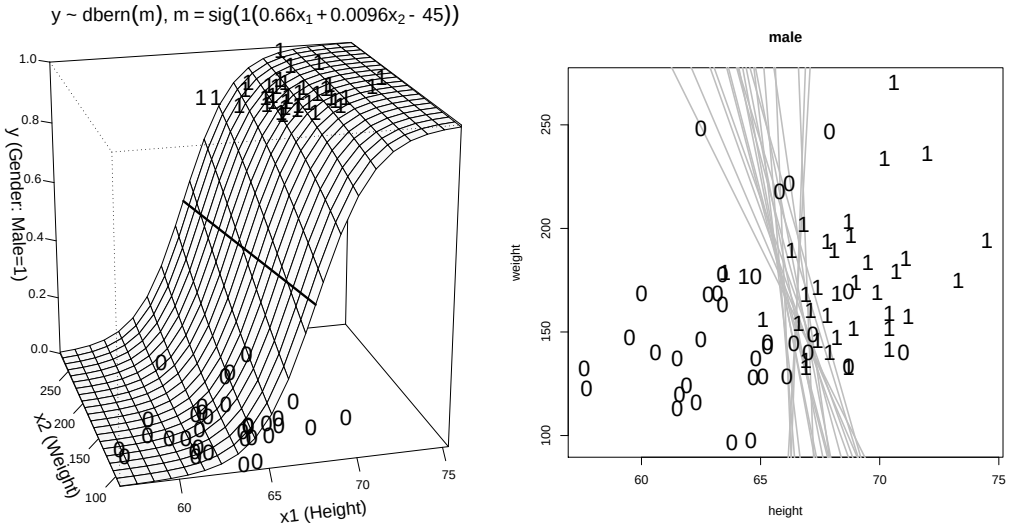

Figure 20.1: Gender (male=1, female=0), height, and weight. Left panel shows data in perspective, with the MLE logistic surface, and $p(y = 1) = .5$ level contour marked as a dark line. The data points are all at $y = 1$ or $y = 0$, not on the logistic surface. Right panel shows the data "from above" with several grey lines that indicate a smattering of credible $p(y = 1) = .5$ level contours derived from Bayesian logistic regression.

20.1 Logistic regression

Suppose we have metric predictors and a dichotomous predicted variable. As a concrete example, suppose we want to predict the gender of a person (coded male=1 and female=0) on the basis of the person's height and weight. Figure 20.1 shows realistic data from 70 people. Males are plotted as numeral 1's, and females are plotted as numeral 0's. You can see that there is notable variation of heights and weights across people, and, importantly, there is considerable overlap in the distributions of males and females. Thus, at any specific combination of height and weight, the best we can do is predict the probability that a person with that height and weight is male.

The model we will use to formulate the probability that $y = 1$, as a function of the two predictors, is the logistic transform of a linear combination of the predictors. Formal details will be reviewed below, but a graphical preview is provided in the left panel of Figure 20.1. The smoothly ascending surface plots the probability $y = 1$ as a function of x_1 and x_2. The parameters of the model control the "cliff", specifically its orientation, location, and steepness.

The maximum likelihood estimate (MLE) of the model parameters provides a single logistic surface that summarizes the distribution of the data. The MLE surface is shown in the left panel of Figure 20.1. Also plotted on the surface, as a dark line, is the level contour at which $p(y = 1) = 0.5$. This level contour marks the points that are halfway up the logistic cliff. The level contour is a convenient summary of the orientation and location of the logistic surface, but the level contour loses information about the steepness of the cliff.

The right panel of Figure 20.1 shows the same data viewed from above. Also plotted are several credible $p(y = 1) = 0.5$ level contours, from a Bayesian analysis. Unlike the single MLE contour in the left panel, the distribution of credible contours reveals our uncertainty

in the parameter estimates of the logistic-regression model. The mechanics of the Bayesian analysis are described in the next sections.

20.1.1 The model

Recall that for multiple linear regression, the central tendency of the predicted value is a linear combination of the predictors, as in Equation 14.3 (p. 307), and as explained at length in Chapter 17. In multiple logistic regression, the linear combination is transformed by a logistic squashing function, and the resulting value, between zero and one, is used as the probability that the predicted value is one. Formally, as in the second row of Table 14.1 on p. 312, logistic regression can be written as

$$\begin{aligned} &= \text{sig}\left({}_0 + {}_1 x_1 + {}_2 x_2 + \cdots \right) \\ y &\sim \text{dbern}( ) \end{aligned}$$

where sig refers to the sigmoid function, which is merely another name for the logistic function:

$$\text{sig}(x) = 1 \quad 1 + \exp(x) \tag{20.1}$$

Graphs of the logistic (a.k.a. sigmoid) function were shown in Figure 14.6 (p. 305) for a single predictor, Figure 14.7 (p. 306) for two predictors, and Figure 14.10 (p. 310) for two predictors with dichotomous predicted values superimposed. Please do review those figures now!

The logit function is the inverse of the logistic function. Formally, for $0 < p < 1$, $\text{logit}(p) = \log(p/(1 \quad p))$. In applications to logistic regression, the "p" is the probability that $y = 1$, and therefore we can write $\text{logit}(p(y=1)) = \log(p(y=1)/p(y=0))$, where the logarithm is the natural logarithm, i.e., the inverse of the exponential function. The ratio, $p(y=1)/p(y=0)$, is called the odds of outcome 1 to outcome 0. The logistic regression model can be written in terms of the logit function, also called the log-odds form, like this:

$$\begin{aligned} \text{logit}( ) &= {}_0 + {}_1 x_1 + {}_2 x_2 + \cdots \\ y &\sim \text{dbern}( ) \end{aligned} \tag{20.2}$$

I prefer the expression in terms of the logistic (i.e., sigmoid) in Equation 20.1 because it provides an explicit specification of that is natural to depict in hierarchical diagrams. But the logit form in Equation 20.2 is useful for interpreting the regression coefficients, as will be described in Section 20.1.3.

The hierarchical diagram for the model is shown in Figure 20.1. It is just like the one for multiple linear regression in Figure 17.4, p. 375, except for the inclusion of a sigmoid and a change in the likelihood function at the bottom of the diagram, from a normal distribution to a Bernoulli distribution.

To reiterate, what the model does is take the i^{th} individual's predictor values, x_{ji}, and generate the probability $_i$ that $y = 1$ for that individual. The Bayesian estimation yields values of the parameters $_0$ and $_j$, that respect the data and the prior.

20.1.2 Doing it in R and BUGS

The implementation of logistic regression in R, BRugs, and JAGS is straightforward, with only a few modifications of the linear regression program presented earlier in the book. The full logistic regression program is listed in Section 20.5.1

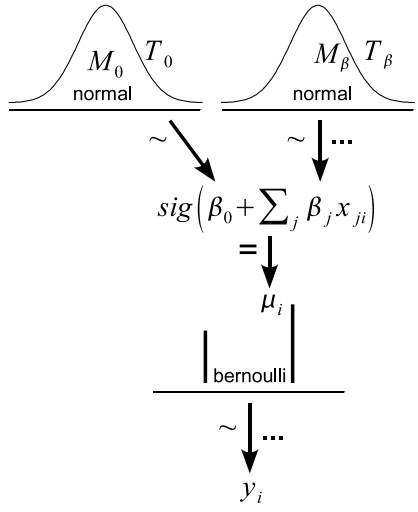

Figure 20.2: Hierarchical model for logistic regression, analogous to the hierarchical model for linear regression in Figure 17.4, p. 375. A prior across regression coefficients can be added, as in Figure 17.6, p. 379.

(MultipleLogisticRegressionBrugs.R). The only notable changes, other than the model, are (i) the use of R's glm function to initialize the chains at the MLE, and (ii) a simple formula for conversion of standardized data to original-scale data. A more thorough explanation is provided before the listing in Section 20.5.1 (MultipleLogisticRegressionBrugs.R).

20.1.3 Interpreting the posterior

Consider again the height, weight, and gender data in Figure 20.1. Aspects of the posterior distribution are shown in Figure 20.3. Often we are primarily interested in knowing how big an influence a predictor has on the predicted value, and, especially, whether that influence is credibly non-zero.

The middle panel of Figure 20.3 indicates that height provides a credibly non-zero predictiveness for gender. The mean value of the coefficient on height is 0.721, which means that the log odds, $\log(p(\text{male})/p(\text{female}))$, increases by 0.721 when height increases by one inch.

This notion of constant increase in log odds is not instantly intuitive. Recall from Equation 20.2 that the regression equation can be expressed in log-odds form as

$$\text{logit}(\ ) = \log\left(\frac{p(y=1)}{p(y=0)}\right) = \ _0 + \ _1 x_1 + \ _2 x_2 + \ldots$$

Therefore, when x_i increases one unit, the log-odds increase $\ _i$ units. Some numerical examples might help. Consider a hypothetical person who weighs 160 pounds. We will compute what happens to the log-odds when the person's height increases by 1 inch. First, consider an increase from 63 inches tall to 64 inches tall. According to the logistic function with parameters set at the mean of the posterior, if the 160-lb person were 63 inches tall, then the probability of being male is 0.0728, and if the person were 64 inches tall, the probability of being male is 0.1390. That increase of 6.62 percentage points corresponds to a change in log odds of $\log(0.1390/(1-0.1390)) - \log(0.0728/(1-0.0728))$, which equals 0.721. Next, consider an increase from 66 inches to 67 inches tall. If the person

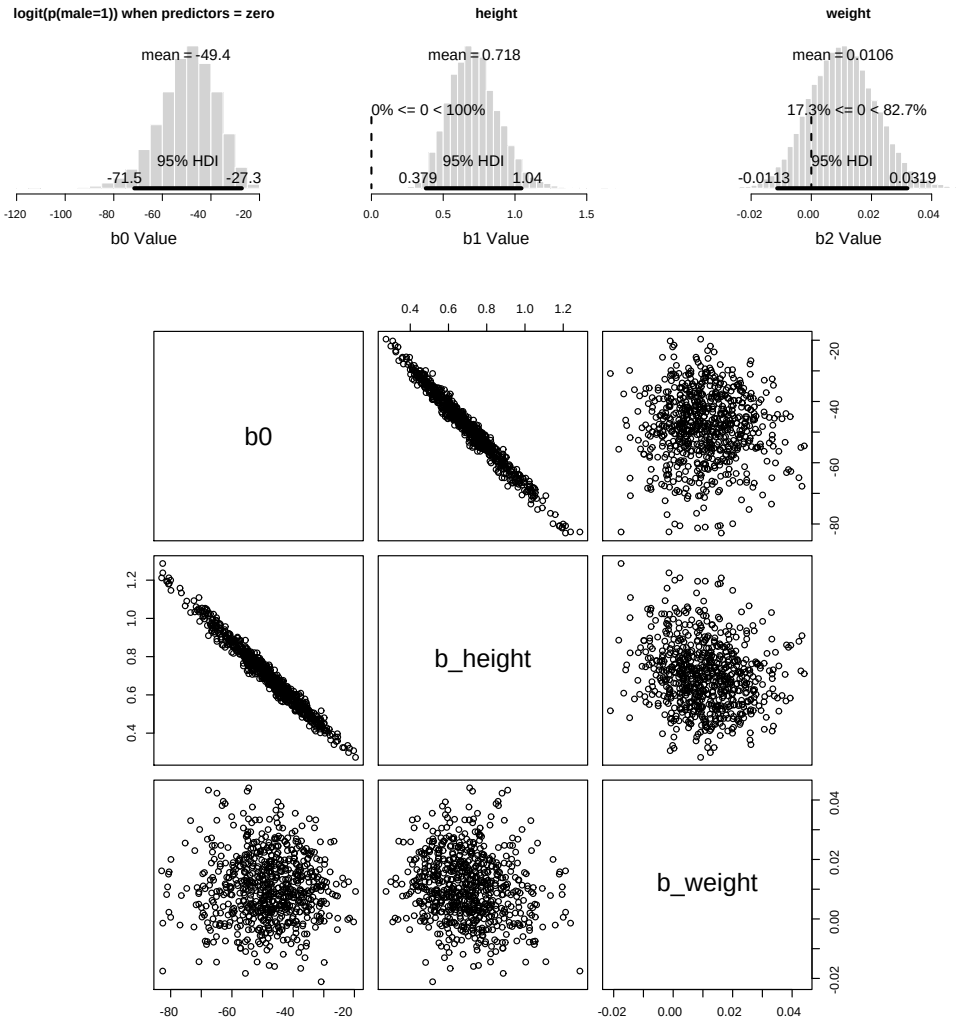

Figure 20.3: Posterior distribution of Bayesian logistic regression of gender on height and weight.

were 66 inches tall, the probability of being male is 0.4056, and if the person were one inch taller, at 67 inches, then the probability of being male is 0.5839. The increase of 17.83 percentage points corresponds to a change in log odds of $\log(0.5839/(1-0.5839))$ $\log(0.4056/(1-0.4056))$, which again equals 0.721. Thus, for any increase of one inch in height, the increase in predicted log-odds of being male is the same, namely 0.721.

The right panel of Figure 20.3 indicates that weight does not provide a credibly non-zero predictiveness for gender, because zero is well among the credible values in the posterior. The mean value of the coefficient on weight is 0.0106, which means that the log odds, $\log(p(\text{male})/p(\text{female}))$, increases by only 0.0106 when weight increases by one pound.

The lower part of Figure 20.3 shows pairwise scatterplots of the 3-dimensional posterior. Notice that the intercept and the coefficient on height are strongly negatively correlated. This correlation results from two properties of the data. First, height is predictive of gender, and therefore the intercept is constrained by height. Second, these original-scale data are not standardized at heights of zero, so if the coefficient on height is increased, the intercept

must be decreased to keep the logistic curve at the appropriate position.

20.1.4 Perils of correlated predictors

When predictors are correlated, care must be taken in interpreting their regression coefficients. This issue was emphasized in Section 17.1.1 in the context of multiple linear regression, but analogous perils arise in logistic regression.

Consider a situation with two predictors. When the predictors are uncorrelated, the posterior will have relatively high certainty regarding the regression coefficients. But when the predictors are strongly correlated, the posterior will have relatively low certainty about the regression coefficients, because many different logistic surfaces can fit the data fairly well. Figure 20.7 in Exercise 20.1 provides an example.

Another peril arising from correlated predictors arises even when one of the correlated predictors is not included in the model. Suppose that one predictor is included in the model, but another predictor is not included, perhaps because the data for the excluded predictor were lost or unobtainable or not even considered in the first place. Suppose further that the true regression coefficient on the included predictor is zero, but the true regression coefficient on the excluded predictor is large positive. The posterior estimate of the included regression coefficient will be non-zero if the included predictor is correlated with the excluded predictor. The reason is intuitively straight forward: As the included predictor increases, the excluded predictor changes (because it is correlated with the included predictor), and therefore the outcome changes, even though the included predictor has no direct predictive value for the outcome. Exercise 20.1 provides an example of this situation.

20.1.5 When there are few 1's in the data

In linear regression, the predicted variable is on a metric scale. Usually the data are distributed over a range of values, without being severely clumped over a single value. In logistic regression, however, the predicted variable is dichotomous, and the data can sometimes consist of mostly 1's or mostly 0's. For example, if the predicted variable is the occurrence of a rare disease, then, by definition, there are few 1's. As another example, the predicted variable might be the occurrence of a defect on a manufacturing assembly line, which is expected to be rare.

The problem with data that have only a few 1's or a few 0's is that the estimate of the regression coefficients is usually relatively uncertain. This makes intuitive sense, because it is only the transition from 0's to 1's that constrains the regression coefficients. Exercise 20.2 provides an example.

20.1.6 Hyperprior across regression coefficients

When there are many candidate predictors, we can use prior knowledge to mutually constrain the estimates of the coefficients. Because all the predictors come from a pool of factors that might have some remotely plausible predictive relation with the dichotomous value, we could reasonably put a hyperprior on the regression coefficients that expresses the assumption that most coefficients are near zero, but some may be notably farther from zero. A distribution that captures this prior knowledge is a t distribution, with a precision that is estimated from the data. This scheme was described in the context of multiple linear regression in Section 17.2, p. 378. A diagram of the hierarchical prior was displayed

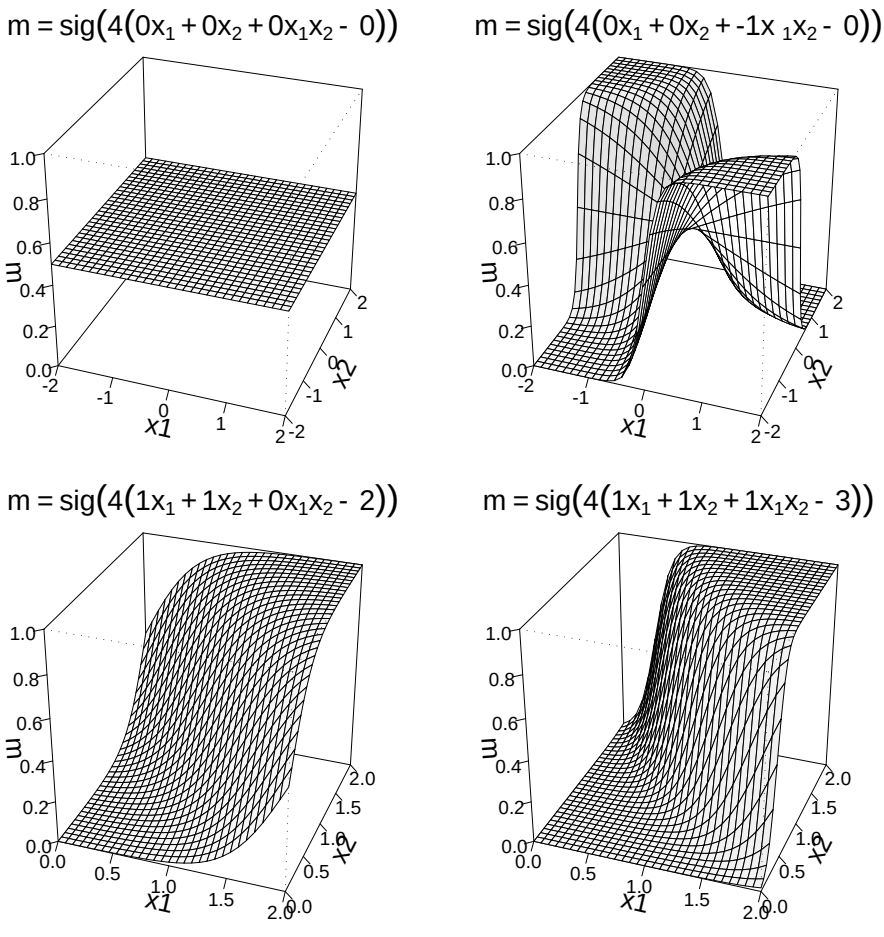

Figure 20.4: Right panels: Logistic regression with interaction. Left panels show corresponding function without any interaction component. Notice that the upper panels include negative values of the predictors, whereas the lower panels do not. In the lower-right graph, when both x_1 and x_2 are sufficiently large negative values (not shown), then the surface rises up toward $m = 1$.

in Figure 17.6, p. 379. The same scheme can be applied to logistic regression, merely by substituting the Bernoulli likelihood function, as in Figure 20.2.

The knowledge expressed in the hyperprior provides shrinkage on the coefficients in logistic regression. When several predictors have estimated coefficients near zero, they drive plausible values of the precision of the overarching prior toward higher values, thereby shrinking estimates of other predictor's coefficients. The shrinkage attenuates false alarms, as discussed at length in Section 17.2.

20.2 Interaction of predictors in logistic regression

Logistic regression can include a multiplicative interaction just as when the predicted variable is metric (recall Figures 14.3 and 17.8). The interpretation of such an interaction requires care.

Suppose that the predicted variable is a two-valued sports fan: Happiness: The respon-

dent says/she is happy or not happy. The two predictors are annual income and healthiness assayed on a metric scale. Intuitively, we can imagine that happiness depends on the conjunction of good health and high income; either one alone may be insu cient for happiness. To put it another way, a person can be unhappy either because of poor health or because of poverty. Thus, the probability of being happy in a conjunctive interaction of health and wealth.

As another example, consider predicting the success of the high-an-air vehicle. One predictor is the intensity of heat under the balloon. A second predictor is the amount of hydrogen in the balloon. Intuitively, the balloon will rise only if there is su cient heat or su cient hydrogen but not both. This is called an exclusive-OR interaction between the predictors.

Figure 20.4 shows graphs of multiplicative interactions of predictors in a logistic function. The top-right panel shows an exclusive-OR: The predicted probability, m, is high only if x_1 is large or x_2 is large but not both. The top-left panel shows absence of an effect as a comparison and reminder that the baseline for a logistic is 0.5.

The lower-right panel of Figure 20.4 shows a conjunctive interaction: The predicted probability is high only if x_1 and x_2 are large. The lower-left panel shows a comparison that has zero interaction. The two lower panels are subtly different: Level contours (not shown) on the zero-interaction cli are straight lines, but level contours (not shown) on the interactive cli are curved, resembling hyperbolas.

Multiplicative interactions can be incorporated into logistic regression models in the same way they are incorporated into metric-predictor models. As was emphasized in the chapter on multiple linear regression, the coefficients on the predictors must be interpreted with care, especially when interactions are involved.

20.3 Logistic ANOVA

There are many situations in which the variable to be predicted is dichotomous, and the predictors are nominal. As an example, recall the habituation-of-condensation experiment introduced in Section 9.3.1, p. 178, and used many times thereafter. The predicted variable was accuracy on each trial, i.e., correct or incorrect. The predictor was a nominal variable that indicated which of four category structures the learner experienced. As another example, recall the relevance-shift experiment of Exercise 13.4, p. 282. Again the predicted variable was accuracy on each trial, and the predictor was a nominal variable indicating which of four relevance shifts the learner experienced. If the predicted variable were metric instead of dichotomous, then these cases could be analyzed with an ANOVA as in Chapter 18.

The logistic function can be used to adapt ANOVA models to dichotomous outcomes. We start with the ANOVA model to generate a predicted tendency for each condition, then pass that value through a sigmoid so it becomes the predicted probability that $y = 1$ for that condition. We also assume that subjects can vary around the condition's predicted probability, just as in the ANOVA model. Figure 20.5 shows the logistic ANOVA model for a single factor, i.e., "oneway" logistic ANOVA. The basic ANOVA model comes directly from the top part of Figure 18.1, p. 403. The beta distribution for subjects within conditions comes directly from the left side of Figure 9.17, p. 183. Notice that this model assumes equal variances for all conditions, like traditional ANOVA, in so far as the same value is used for all groups simultaneously.

Figure 20.6 shows the results of running logistic oneway ANOVA on the Ultra-

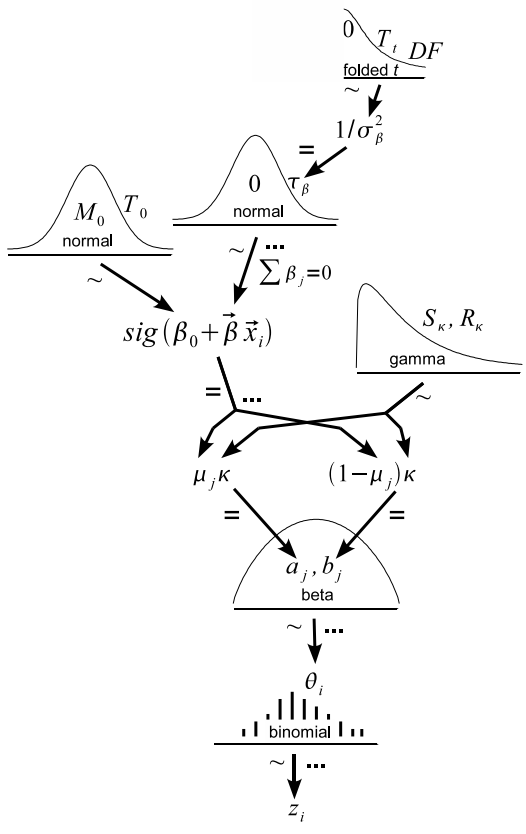

Figure 20.5: Bayesian logistic ANOVA. The ANOVA model, in the upper parts, generates the $_j$ for the groups through the sigmoidal transform. The beta distribution, in the lower part of the diagram, describes the within-group variability across subjects. The same is assumed for all groups, which is analogous to homogeneity of variance in metric ANOVA.

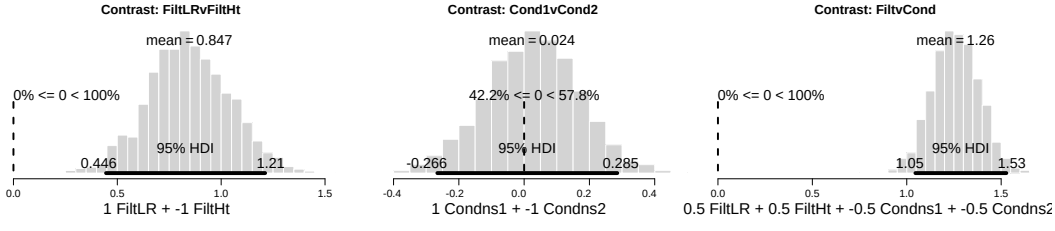

Figure 20.6: Results of logistic ANOVA applied to ltration/condensation experiment. Compare with Figure 9.16, p. 181, which did not use an ANOVA model for these data. The scale on these histograms is rence of ANOVA-model $_j$ values, not di erence of $_c$ values as in Figure 9.16.

tion/condensation data. The comparisons are differences of the parameters in the ANOVA model. The parameters are not probabilities, but the parameters can be mapped to probabilities via the logistic function. The comparisons of the parameters from different conditions yield the same conclusions as the analysis from each of the chapters, as in Figure 9.16, p. 181, which did not use an ANOVA model for these data.

If the ANOVA model leads to the same conclusions as the nonANOVA approach, what are the advantages or disadvantages of the two models? An advantage of the logistic ANOVA model is that it tends to have larger autocorrelation in its parameters, and therefore takes longer to run because more thinning is needed. And the disadvantage of the logistic ANOVA model is that the parameters are not instantly interpretable as probabilities, but must be converted through the logistic. But the ANOVA approach has great advantages when the situation to be modeled becomes more complex. In particular, when there are two or more predictors, the ANOVA model provides a natural representation for the main effects and interactions of the predictors. This generalizability of the logistic ANOVA model is its trump card.

Some analysts of dichotomous data will treat z_i/N_i as if it were a metric value, and apply standard ANOVA. Because standard ANOVA assumes that the data come from normal distributions, the analyst will transform the values to something like $2\sin^{-1}\sqrt{z_i/N_i}$ (e.g., Winer, Brown, & Michels, 1991, p. 356), to approximate normality. Unfortunately the approximation is not very good for small N or for values of z_i/N_i that are very close to 0 or 1. Most importantly, the ratio z_i/N_i loses information about the magnitude of N. If all we know is that $z_i/N_i = 0.75$, we don't know whether $z_i = 3$ and $N_i = 4$, or $z_i = 3000$ and $N_i = 4000$. This matters because a proportion based on a large sample should influence our beliefs more than a proportion based on a small sample. Therefore, converting to proportions fails in unbalanced designs, i.e., when there are different sample sizes for different subjects. In the Bayesian approach used here, every individual observation has an equal influence, and the Bayesian analysis automatically handles unbalanced designs appropriately.

20.3.1 Within-subject designs

Fortunately, there is no news when it comes to repeated measures, i.e., within-subject designs. All the concepts of Section 19.2, p. 432, apply the same way to logistic ANOVA. There were two main messages of that section. First, in terms of the structure, if every subject contributes many observations to every combinations of conditions, then it makes sense to apply a logistic ANOVA model to every individual subject, with hyperdistributions on the ANOVA coefficients to allow mutually informed estimation across subjects. If every subject contributes only one observation to any combination of conditions, then it makes sense to include a subject factor in the model, with no interaction of that subject factor with the other factors. Second, in terms of whether to use a repeated measures design at all, there must be a good rationale for assuming that repeated measures from the same subject are reasonably independent of each other, so that any of these designs can be applied.

20.4 Summary

The extension of linear regression and ANOVA to logistic regression and logistic ANOVA is straight forward. The hierarchical models of Figures 20.2 and 20.5 illustrated how the core linear model of regression or ANOVA is passed through a logistic function, and the

resulting value is used as the mean for a Bernoulli (or binomial) likelihood function. All the basic structural concepts of linear regression and ANOVA apply to logistic regression and ANOVA.

Only a few new technical concepts introduced in this chapter. One was the notion of level contour on the logistic surface. The contour at level .5 was useful for marking the boundary between x values for which $p(y=1) < .5$ and x values for which $p(y=1) > .5$.

Another new technical concept was interpreting a coefficient in logistic regression in terms of the increase in log-odds of the outcome when the predictor increases by one unit. Although it was not previously mentioned, the same idea applies to logistic ANOVA: The β_j value indicates the increase in log-odds of the outcome relative to the baseline, β_0. This interpretation must be qualified, however, when the model includes interaction terms, whether it is regression or ANOVA.

The remaining technical issues arise only in the mechanics of computing the analyses. The next section, which includes the R code, explains these details. There are two main issues, one regarding standardizing the predictors (for regression only) and transforming the parameters back to the original scales, and the other regarding how to initialize the chains by starting with the maximum likelihood estimate (MLE).

20.5 R code

20.5.1 Logistic regression code

The program for logistic regression is much like the analogous program for linear regression, with only a few changes. One obvious change is that the likelihood function is Bernoulli, not normal. Beyond the model specification, it is important to understand the following details.

The predictor values are standardized, to reduce autocorrelation in MCMC sampling. Notice, for example, the highly correlated original-scale parameter values in Figure 20.3, which would be difficult to navigate by most MCMC sampling algorithms. Unlike linear regression, however, the predicted values are not standardized, because the predicted values must by zero or one. Because the predicted values are not standardized, the conversion of the standardized parameter values to original-scale parameter values is less complicated:

$$\begin{aligned}
\text{logit}(\mu_i) &= \beta_0 + \sum_j \beta_j z_{ji} \\
&= \beta_0 + \sum_j \beta_j \frac{x_{ji} - E[x]_j}{s_{x_j}} \\
&= \underbrace{\beta_0 - \sum_j \beta_j E[x]_j / s_{x_j}}_{\beta_0} + \sum_j \underbrace{\beta_j / s_{x_j}}_{\beta_j} x_{ji}
\end{aligned}$$

This conversion is executed at every step in the MCMC chain, and is performed at the end of the program.

Initialization of the chains is based on parameter values that are plausible given the data. Specifically, the values of the MLE are used, because R has a built-in function for determining the MLE of logistic regression, namely the `glm` function with the `family` argument specified as `binomial(logit)`.

The final lines of the program perform an MLE of the logistic regression, just for comparison with the Bayesian results. The MLE is typically around the most credible values in the Bayesian posterior. But the posterior provides more information about correlations of parameters and the range of uncertainty in each parameter.

(MultipleLogisticRegressionBrugs.R)

```
graphics.off()
rm(list=ls(all=TRUE))
fname = "MultipleLogisticRegressionBrugs"
library(BRugs)              # Kruschke, J. K. (2010). Doing Bayesian data analysis:
                              # A Tutorial with R and BUGS. Academic Press / Elsevier.
#------------------------------------------------- ----------------------------
# THE MODEL.
modelstring = "
# BUGS model specification begins here...
model {
  for( i in 1 : nData ) {
    y[i] ~ dbern( mu[i] )
    mu[i] <- 1/(1+exp(-( b0 + inprod( b[] , x[i,] ))))
  }
  b0 ~ dnorm( 0 , 1.0E-12 )
  for ( j in 1 : nPredictors ) {
    b[j] ~ dnorm( 0 , 1.0E-12 )
  }
}
# ... end BUGS model specification
" # close quote for modelstring
# Write model to a file:
writeLines(modelstring,con="model.txt")
modelCheck( "model.txt" )

#------------------------------------------------- ----------------------------
# THE DATA.

dataSource = c( "HtWt" , "Cars" , "HeartAttack" )[3]

if ( dataSource == "HtWt" ) {
  fname = paste( fname , dataSource , sep="" )
  # Generate random but realistic data:
  source( "HtWtDataGenerator.R" )
  dataMat = HtWtDataGenerator( nSubj = 70 , rndsd=474 )
  predictedName = "male"
  predictorNames = c( "height" , "weight" )
  nData = NROW( dataMat )
  y = as.matrix( dataMat[,predictedName] )
  x = as.matrix( dataMat[,predictorNames] )
  nPredictors = NCOL( x )
}

if ( dataSource == "Cars" ) {
  fname = paste( fname , dataSource , sep="" )
  dataMat = read.table(file="Lock1993data.txt",header=T ,sep=" ")
  predictedName = "AirBag"
  predictorNames = c( "MidPrice" , "RPM" , "Uturn" )
  nData = NROW( dataMat )
  y = as.matrix( as.numeric( dataMat[,predictedName] > 0 ) ) #  0,1,2 to 0,1
  x = as.matrix( dataMat[,predictorNames] )
  nPredictors = NCOL( x )
```

```
}

if ( dataSource == "HeartAttack" ) {
  fname = paste( fname , dataSource , sep="" )
  dataMat = read.table(file="BloodDataGeneratorOutput.t        xt",header=T,sep="  ")
  predictedName = "HeartAttack"
  predictorNames = c( "Systolic", "Diastolic", "Weight", "C            holesterol",
                      "Height", "Age" )
#  predictorNames = c( "Systolic", "Diastolic" )
  nData = NROW( dataMat )
  y = as.matrix( dataMat[,predictedName] )
  x = as.matrix( dataMat[,predictorNames] )
  nPredictors = NCOL( x )
}

# Prepare data for BUGS:
# Re-center data at mean, to reduce autocorrelation in MCMC s    ampling.
# Standardize (divide by SD) to make initialization easier.
standardizeCols = function( dataMat ) {
    zDataMat = dataMat
    for ( colIdx in 1:NCOL( dataMat ) ) {
        mCol = mean( dataMat[,colIdx] )
        sdCol = sd( dataMat[,colIdx] )
        zDataMat[,colIdx] = ( dataMat[,colIdx] - mCol ) / sdCol
    }
    return( zDataMat )
}
zx = standardizeCols( x )
zy = y  # y is not standardized; must be 0,1

# Get the data into BUGS:
datalist = list(
            x = zx ,
            y = as.vector( zy ) , # BUGS does not treat 1-column mat as vecto    r
            nPredictors = nPredictors ,
            nData = nData
)
modelData( bugsData( datalist ) )


#------------------------------------------------                    ----------------------------
# INTIALIZE THE CHAINS.

nchain = 3
modelCompile( numChains = nchain )

genInitList <- function() {
    glmInfo = glm( datalist$y ~ datalist$x , family=binomial(l          ogit) ) # R func.
    show( glmInfo ) ; flush.console() # display in case glm() has          troubles
    b0Init = glmInfo$coef[1]
    bInit = glmInfo$coef[-1]
    return( list(
        b0 = b0Init ,
        b = bInit
    ) )
}
for ( chainIdx in 1 : nchain ) {
    modelInits( bugsInits( genInitList ) )
}
```

```
112
113 #------------------------------------------------                    ----------------------------
114 # RUN THE CHAINS
115
116 # burn in
117 BurnInSteps = 1000
118 modelUpdate( BurnInSteps )
119 # actual samples
120 samplesSet( c( "b0" , "b" ) )
121 stepsPerChain = ceiling(5000/nchain)
122 thinStep = 50  # check autocorrelation and increase as needed
123 modelUpdate( stepsPerChain , thin=thinStep )
124
125 #------------------------------------------------                    ----------------------------
126 # EXAMINE THE RESULTS
127
128 source("plotChains.R")
129 source("plotPost.R")
130
131 # Check chains for mixing
132 checkConvergence = T
133 if ( checkConvergence ) {
134   b0Sum = plotChains( "b0" , saveplots=F , filenameroot=fnam      e )
135   bSum  = plotChains( "b"  , saveplots=F , filenameroot=fname )
136 }
137
138 # Extract chain values:
139 zb0Sample = matrix( samplesSample( "b0" ) )
140 chainLength = length(zb0Sample)
141 zbSample = NULL
142 for ( j in 1:nPredictors ) {
143     zbSample = cbind( zbSample , samplesSample( paste("b[",j,      "]",sep="") ) )
144 }
145
146 # Convert to original scale:
147 x = dataMat[,predictorNames]
148 y = dataMat[,predictedName]
149 My = mean(y)
150 SDy = sd(y)
151 Mx = apply(x,2,mean)
152 SDx = apply(x,2,sd)
153 b0Sample = 0 * zb0Sample
154 bSample = 0 * zbSample
155 for ( stepIdx in 1:chainLength ) {
156       b0Sample[stepIdx] = ( zb0Sample[stepIdx]
157                                   - sum( Mx / SDx * zbSample[stepIdx,] ) )
158       for ( j in 1:nPredictors ) {
159         bSample[stepIdx,j] = zbSample[stepIdx,j] / SDx[j]
160       }
161 }
162
163 # Examine sampled values, z scale:
164 windows()
165 thinIdx = ceiling(seq(1,chainLength,length=700))
166 pairs(   cbind( zb0Sample[thinIdx] , zbSample[thinIdx,] )   ,
167           labels=c( "zb0", paste("zb",predictorNames,sep="") ) )
168 # Examine sampled values, original scale:
169 windows()
170 pairs( cbind( b0Sample[thinIdx] , bSample[thinIdx,] ) ,
```

```
171           labels=c( "b0", paste("b_",predictorNames,sep="") ) )
172 dev.copy2eps(file=paste(fname,"PostPairs.eps",sep="          "))
173
174 # Display the posterior :
175 windows(3.5*(1+nPredictors),2.75)
176 layout( matrix(1:(1+nPredictors),nrow=1) )
177 histInfo = plotPost( b0Sample , xlab="b0 Value" , compVal=N      ULL , breaks=30 ,
178                          main=paste( "logit(p(", predictedName ,
179                                      "=1)) when predictors = zero" , sep="" ) )
180 for ( bIdx in 1:nPredictors ) {
181 histInfo = plotPost( bSample[,bIdx] , xlab=paste("b",bId            x," Value",sep="") ,
182                          compVal=0.0 , breaks=30 ,
183                          main=paste(predictorNames[bIdx]) )
184 }
185 dev.copy2eps(file=paste(fname,"PostHist.eps",sep=""          ))
186
187 # Plot data with .5 level contours of believable logistic sur             faces.
188 # The contour lines are best interpreted when there are only t        wo predictors.
189 for ( p1idx in 1:(nPredictors-1) ) {
190   for ( p2idx in (p1idx+1):nPredictors ) {
191     windows()
192     xRange = range(x[,p1idx])
193     yRange = range(x[,p2idx])
194     # make empty plot
195     plot( NULL , NULL , main=predictedName , xlim=xRange , ylim=  yRange ,
196           xlab=predictorNames[p1idx] , ylab=predictorNames[p2id        x] )
197     # Some of the 50% level contours from the posterior sample.
198     for ( chainIdx in ceiling(seq( 1 , chainLength , length=20 ))             ) {
199       abline( -( b0Sample[chainIdx]
200                  + if (nPredictors>2) {
201                      bSample[chainIdx,c(-p1idx,-p2idx)]*Mx[c(-p1idx,-p2i            dx)]
202                    } else { 0 } )
203                  / bSample[chainIdx,p2idx] ,
204               -bSample[chainIdx,p1idx]/bSample[chainIdx,p2idx] ,
205               col="grey" , lwd = 2 )
206     }
207     # The data points:
208     for ( yVal in 0:1 ) {
209       rowIdx = ( y == yVal )
210       points( x[rowIdx,p1idx] , x[rowIdx,p2idx] , pch=as.chara            cter(yVal) ,
211               cex=1.75 )
212     }
213     dev.copy2eps(file=paste(fname,"PostContours",p1idx,         p2idx,".eps",sep=""))
214   }
215 }
216
217 #------------------------------------------------                          ----------------------------
218
219 # MLE logistic regression:
220 glmRes = glm( datalist$y ~ as.matrix(x) , family=binomial(          logit) )
221 show( glmRes )
```

20.5.2 Logistic ANOVA code

This program implements the model shown in Figure 20.5.

(LogisticOnewayAnovaBrugs.R)

```
1 graphics.off()
```

```
rm(list=ls(all=TRUE))
fnroot = "LogisticOnewayAnovaBrugs"
library(BRugs)           # Kruschke, J. K. (2010). Doing Bayesian dat  a analysis:
                          # A Tutorial with R and BUGS. Academic Press / Elsevier.
#------------------------------------------------            ----------------------------
# THE MODEL.

modelstring = "
# BUGS model specification begins here...
model {
  for ( i in 1:Ntotal ) {
    z[i] ~ dbin( theta[i] , n[i] )
    theta[i] ~ dbeta( aBeta[x[i]] , bBeta[x[i]] )I(0.001,0.99          9)
  }
  for ( j in 1:NxLvl ) {
    aBeta[j] <- mu[j] * k
    bBeta[j] <- (1-mu[j]) * k
    mu[j] <- 1 / ( 1 + exp( -( a0 + a[j] ) ) )
    a[j] ~ dnorm( 0.0 , atau )
  }
  k ~ dgamma( 1.0 , 0.01 )
  a0 ~ dnorm( 0 , 0.001 )
  atau <- 1 / pow( aSD , 2 )
  aSD <- abs( aSDunabs ) + .1
  aSDunabs ~ dt( 0 , 0.001 , 2 )
}
# ... end BUGS model specification
" # close quote for modelstring
# Write model to a file, and send to BUGS:
writeLines(modelstring,con="model.txt")
modelCheck( "model.txt" )

#------------------------------------------------            ----------------------------
# THE DATA.

# Specify data source:
dataSource = c( "Filcon" , "Relshift" , "Random" )[1]
# Load the data:

sigmoid = function( x ) { return( 1 / ( 1 + exp( -x ) ) ) }
logit = function( y ) { return( log( y / (1-y) ) ) }

if ( dataSource == "Filcon" ) {
  fnroot = paste( fnroot , dataSource , sep="" )
  x = c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1       ,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2
  n = c(64,64,64,64,64,64,64,64,64,64,64,64,64,64,64,64     ,64,64,64,64,64,64,64,64,64,64,64,64,64,64
  z = c(45,63,58,64,58,63,51,60,59,47,63,61,60,51,59,45     ,61,59,60,58,63,56,63,64,64,60,64,62,49,64
  Ntotal = length(z)
  xnames = c("FiltLR","FiltHt","Condns1","Condns2")
  NxLvl = length(unique(x))
  contrastList = list( FiltLRvFiltHt = c(1,-1,0,0) ,
                       Cond1vCond2 = c(0,0,1,-1) ,
                       FiltvCond = c(1/2,1/2,-1/2,-1/2) )
}

if ( dataSource == "Relshift" ) {
  fnroot = paste( fnroot , dataSource , sep="" )
  #source( "Kruschke1996CSdata.R" ) # if it has not yet been ru   n
  load("Kruschke1996CSdatsum.Rdata") # loads CondOfSubj,  nCorrOfSubj, nTrlOfSubj
```

```
  x = CondOfSubj
  n = nTrlOfSubj
  z = nCorrOfSubj
  Ntotal = length(z)
  xnames = c("Rev","Rel","Irr","Cmp")
  NxLvl = length(unique(x))
  contrastList = list( REVvREL = c(1,-1,0,0) , RELvIRR = c(0,1 ,-1,0) ,
                       IRRvCMP = c(0,0,1,-1) , CMPvOneRel = c(0,-1/2,-1/2,1) ,
                       FourExvEightEx = c(-1,1/3,1/3,1/3) ,
                       OneRelvTwoRel = c(-1/2,1/2,1/2,-1/2) )
}

if ( dataSource == "Random" ) {
  fnroot = paste( fnroot , dataSource , sep="" )
  #set.seed(47405)
  a0true = -0.5
  atrue = c( 0.8 , -0.3 , -0.5 ) # sum to zero
  ktrue = 100
  subjPerCell = 50
  nPerSubj = 100
  datarecord = matrix( 0, ncol=3 , nrow=length(atrue)*subjP erCell )
  colnames(datarecord) = c("x","z","n")
  rowidx = 0
  for ( xidx in 1:length(atrue) ) {
    for ( subjidx in 1:subjPerCell ) {
      rowidx = rowidx + 1
      datarecord[rowidx,"x"] = xidx
      mu = sigmoid(a0true+atrue[xidx])
      theta = rbeta( 1 , mu*ktrue , (1-mu)*ktrue )
      datarecord[rowidx,"z"] = rbinom( 1 , prob=theta , size=nPe rSubj )
      datarecord[rowidx,"n"] = nPerSubj
    }
  }
  datarecord = data.frame( x=as.factor(datarecord[,"x"]) , z=datarecord[,"z"] ,
                           n=datarecord[,"n"] )
  z = as.numeric(datarecord$z)
  Ntotal = length(z)
  n = as.numeric(datarecord$n)
  x = as.numeric(datarecord$x)
  xnames = levels(datarecord$x)
  NxLvl = length(unique(x))
  # Construct list of all pairwise comparisons, to compare wit h NHST TukeyHSD:
  contrastList = NULL
  for ( g1idx in 1:(NxLvl-1) ) {
    for ( g2idx in (g1idx+1):NxLvl ) {
      cmpVec = rep(0,NxLvl)
      cmpVec[g1idx] = -1
      cmpVec[g2idx] = 1
      contrastList = c( contrastList , list( cmpVec ) )
    }
  }
}

# Specify the data in a form that is compatible with BRugs mode l, as a list:
datalist = list(
  z = z ,
  n = n ,
  x = x ,
  Ntotal = Ntotal ,
```

```
120     NxLvl = NxLvl
121   )
122   # Get the data into BRugs:
123   modelData( bugsData( datalist ) )
124
125   #------------------------------------------------                    ----------------------------
126   # INTIALIZE THE CHAINS.
127
128   # Autocorrelation within chains is large, so use several cha          ins to reduce
129   # degree of thinning. But we still have to burn-in all the chai          ns, which takes
130   # more time with more chains.
131   nchain = 10
132   modelCompile( numChains = nchain )
133
134   if ( F ) {
135       modelGenInits() # often won't work for diffuse prior
136   } else {
137     #   initialization based on data
138     theData = data.frame( pr=.01+.98*datalist$z/datalist$n              ,
139                                 x=factor(x,labels=xnames) )
140     a0 = mean( logit(theData$pr) )
141     a = aggregate( logit(theData$pr) , list( theData$x ) , mean )          [,2] - a0
142     mGrp = aggregate( theData$pr , list( theData$x ) , mean )[,2]
143     sdGrp = aggregate( theData$pr , list( theData$x ) , sd )[,2]
144     kGrp = mGrp*(1-mGrp)/sdGrp^2 - 1
145     k = mean(kGrp)
146     genInitList <- function() {
147       return(
148             list(
149                   a0 = a0 ,
150                   a = a ,
151                   aSDunabs = sd(a) ,
152                   theta = theData$pr ,
153                   k = k
154             )
155       )
156     }
157     for ( chainIdx in 1 : nchain ) {
158       modelInits( bugsInits( genInitList ) )
159     }
160   }
161
162   #------------------------------------------------                    ----------------------------
163   # RUN THE CHAINS
164
165   # burn in
166   BurnInSteps = 10000
167   modelUpdate( BurnInSteps )
168   # actual samples
169   samplesSet( c( "a0" ,   "a" , "aSD" , "k" ) )
170   stepsPerChain = ceiling(2000/nchain)
171   thinStep = 750
172   modelUpdate( stepsPerChain , thin=thinStep )
173
174   #------------------------------------------------                    ----------------------------
175   # EXAMINE THE RESULTS
176
177   source("plotChains.R")
178   source("plotPost.R")
```

```
179
180 checkConvergence = T
181 if ( checkConvergence ) {
182     sumInfo = plotChains( "a0" , saveplots=F , filenameroot=fn        root )
183     sumInfo = plotChains( "a" , saveplots=F , filenameroot=fnr        oot )
184     sumInfo = plotChains( "aSD" , saveplots=F , filenameroot=f        nroot )
185     sumInfo = plotChains( "k" , saveplots=F , filenameroot=fnr        oot )
186 }
187
188 # Extract and plot the SDs:
189 aSDSample = samplesSample("aSD")
190 windows()
191 par( mar=c(3,1,2.5,0) , mgp=c(2,0.7,0) )
192 histInfo = plotPost( aSDSample , xlab="aSD" , main="a SD" , b     reaks=30 )
193 dev.copy2eps(file=paste(fnroot,"SD.eps",sep=""))
194
195 # Extract a values:
196 a0Sample = samplesSample( "a0" )
197 chainLength = length(a0Sample)
198 aSample = array( 0 , dim=c( datalist$NxLvl , chainLength ) )
199 for ( xidx in 1:datalist$NxLvl ) {
200     aSample[xidx,] = samplesSample( paste("a[",xidx,"]",se          p="") )
201 }
202
203 # Convert to zero-centered b values:
204 mSample = array( 0, dim=c( datalist$NxLvl , chainLength ) )
205 for ( stepIdx in 1:chainLength ) {
206     mSample[,stepIdx ] = ( a0Sample[stepIdx] + aSample[,stepI      dx] )
207 }
208 b0Sample = apply( mSample , 2 , mean )
209 bSample = mSample - matrix(rep( b0Sample ,NxLvl),nrow=NxL vl,byrow=T)
210
211 # Plot b values:
212 windows(datalist$NxLvl*2.75,2.5)
213 layout( matrix( 1:datalist$NxLvl , nrow=1 ) )
214 par( mar=c(3,1,2.5,0) , mgp=c(2,0.7,0) )
215 for ( xidx in 1:datalist$NxLvl ) {
216     plotPost( bSample[xidx,] , breaks=30 ,
217               xlab=bquote(beta[.(xidx)]) ,
218               main=paste(xnames[xidx])   )
219 }
220 dev.copy2eps(file=paste(fnroot,"b.eps",sep=""))
221
222 # Consider parameter correlations:
223 kSample = samplesSample("k")
224 windows()
225 pairs( cbind( b0Sample , t(bSample) , kSample ) , labels=c("       b0",xnames,"k") )
226
227 # Display contrast analyses
228 nContrasts = length( contrastList )
229 if ( nContrasts > 0 ) {
230     nPlotPerRow = 5
231     nPlotRow = ceiling(nContrasts/nPlotPerRow)
232     nPlotCol = ceiling(nContrasts/nPlotRow)
233     windows(3.75*nPlotCol,2.5*nPlotRow)
234     layout( matrix(1:(nPlotRow*nPlotCol),nrow=nPlotRow,n       col=nPlotCol,byrow=T) )
235     par( mar=c(4,0.5,2.5,0.5) , mgp=c(2,0.7,0) )
236     for ( cIdx in 1:nContrasts ) {
237         contrast = matrix( contrastList[[cIdx]],nrow=1) # make it          a row matrix
```

```
238           incIdx = contrast!=0
239           histInfo = plotPost( contrast %*% bSample , compVal=0 , brea     ks=30 ,
240                       xlab=paste( round(contrast[incIdx],2) , xnames[incIdx]           ,
241                                   c(rep("+",sum(incIdx)-1),"") , collapse=" " ) ,
242                       cex.lab = 1.5 ,
243                       main=paste( "Contrast:", names(contrastList)[cIdx] ) )
244       }
245       dev.copy2eps(file=paste(fnroot,"xContrasts.eps",sep           =""))
246   }
247
248   #=============================================================================
249   # Do NHST ANOVA:
250
251   theData = data.frame( y=z/n , x=factor(x,labels=xnames) )
252   aovresult = aov( y ~ x , data = theData )
253   cat("\n-------------------------------------------                    ----------------------\n\n")
254   print( summary( aovresult ) )
255   cat("\n-------------------------------------------                    ----------------------\n\n")
256   print( model.tables( aovresult , "means" ) , digits=4 )
257   windows()
258   boxplot( y ~ x , data = theData )
259   cat("\n-------------------------------------------                    ----------------------\n\n")
260   print( TukeyHSD( aovresult , "x" , ordered = FALSE ) )
261   windows()
262   plot( TukeyHSD( aovresult , "x" ) )
263   if ( F ) {
264     for ( xIdx1 in 1:(NxLvls-1) ) {
265       for ( xIdx2 in (xIdx1+1):NxLvls ) {
266         cat("\n-------------------------------------------                        ----------\n\n")
267         cat( "xIdx1 = " , xIdx1 , ", xIdx2 = " , xIdx2 ,
268              ", M2-M1 = " , mean(score[x==xIdx2])-mean(score[x==xIdx1     ]) ,
269              "\n" )
270         print( t.test( score[ x == xIdx2 ] , score[ x == xIdx1 ] ) )
271       }
272     }
273   }
274   cat("\n-------------------------------------------                    ----------------------\n\n")
275
276   #=============================================================================
```

20.6 Exercises

Exercise 20.1. [Purpose: Correlated predictors in logistic regression.] For this exercise, we'll use some ctitious data regarding whether or not a patient had a heart attack within a year after a check up that included measurements of systolic and diastolic blood pressures, cholesterol, weight, height, and age. The data are completely fabricated and ctional, for illustrative purposes only. The data are generated by the program BloodDataGenerator.R , which is available at the author's web site.

(A) For this part, we will generate two data sets. In one set, the predictors have correlations of zero with each other. In the second set, the two predictors have a very strong positive correlation, but still zero correlation with all the other predictors. For both sets, the true regression coefficient on the first predictor is zero, but the true regression coefficient on the the predictor is positive. Use the program BloodDataGenerator.R to generate these two data sets, by selecting the relevant correlation matrix at the top of the pro-

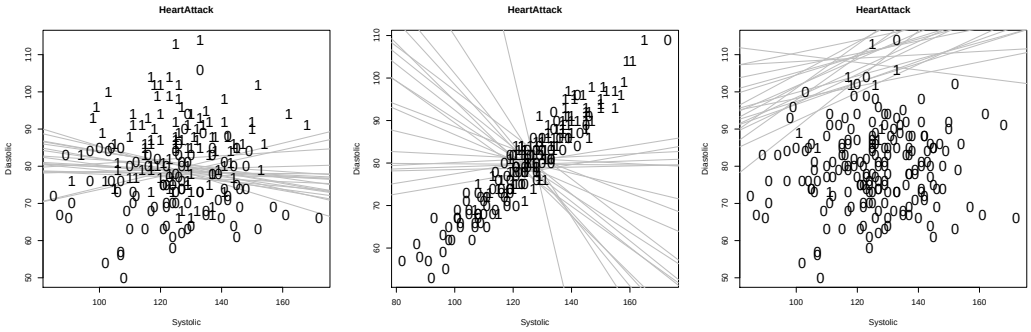

Figure 20.7: In all panels, the true regression coefficient on Systolic is zero, and the true regression coefficient on Diastolic is positive (non-zero). In the left panel, the predictors are uncorrelated, and the credible $p = .5$ level contours are fairly certain. In the middle panel, the predictors are highly correlated, and the credible $p = .5$ level contours are very uncertain. In the right panel, only a small percentage of the data has value 1, and again the level contours are very uncertain. See Exercises 20.1 and 20.2.

gram. For each data set, conduct a Bayesian analysis using the program in Section 20.5.1 (MultipleLogisticRegressionBrugs.R). Be sure to specify that the only predictors are the first two, namely systolic and diastolic pressure. Include the posterior contour plots, which should look similar to those in Figure 20.7. Be sure also to include the histograms of the marginal posterior distributions. In which case are the HDIs narrower?

(B) For this part, we generate a different fictitious data set, in which the first two predictors (systolic and diastolic) have zero correlation with each other, but they are correlated with another predictor (weight). The first predictor (systolic) has a true regression coefficient of zero, but the second (diastolic) and third (weight) predictors have positive regression coefficients. Run the Bayesian logistic regression analysis using only the first two predictors. What is the estimate of the regression coefficient on the first predictor? Include the histograms of the marginals of the posterior. Now re-run the Bayesian logistic regression analysis including all six predictors. What is the estimate of the regression coefficient on the first predictor? Include the histograms of the marginals of the posterior (for which you might need to manually stretch the graph window or change the character sizes in the plot). Why are the estimates of the regression coefficient for the first predictor different in the two analyses?

Exercise 20.2. [Purpose: Small proportion of 1's in logistic regression.] For this exercise, we again use fictitious data generated by the program BernGoodDataGenerator.R . Set all the inter-predictor correlations to zero, by selecting the appropriate correlation matrix at the top of the program. Change the *proportionOnes* constant to 0.02. This causes the proportion of 1's in the data to be very small. In other words, the threshold for the sigmoidal function is set very high, so that the weighted sum of the predictors must be very large before $p(y = 1)$ gets very large. This sort of situation, in which there are few 1's, is not unusual in real data. In heart attack data, for example, there will be few people who have a heart attack in the year following a routine check up. Run a Bayesian logistic regression on the data, using only the first two predictors. What are the estimates of the regression coefficients? Include the histograms of the marginals of the posterior, and include the posterior contour plot, which should look similar to the right panel of Figure 20.7. Compare the widths of the HDI's to

the results of the first part of the previous exercise.

Exercise 20.3. [Purpose: Logistic ANOVA without assuming homogeneity of variance.] Extend the program of Section 20.5.2 (LogisticOnewayAnovaBrugs.R) so that it allows different certainty () parameters for each condition, with a hyperdistribution across conditions, just as in the model on the right side of Figure 9.17, p. 183. Apply the model to the filtration-condensation data, and compare the results with the results of Exercise 9.2, p. 191. If you already did Exercise 18.3, you can simply modify your program for that Exercise. Programming hint: Here's what the model specification might look like: (LogisticOnewayAnovaHeteroVarBrugs.R)

```
model {
   for ( i in 1:Ntotal ) {
      z[i] ~ dbin( theta[i] , n[i] )
      theta[i] ~ dbeta( aBeta[x[i]] , bBeta[x[i]] )I(0.001,0.999)
   }
   for ( j in 1:NxLvl ) {
      aBeta[j] <- mu[j] * k[j]
      bBeta[j] <- (1-mu[j]) * k[j]
      mu[j] <- 1 / ( 1 + exp( -( a0 + a[j] ) ) )
      a[j] ~ dnorm( 0.0 , atau )
      k[j] ~ dgamma( skappa , rkappa )
   }
   a0 ~ dnorm( 0 , 0.001 )
   atau <- 1 / pow( aSD , 2 )
   aSD <- abs( aSDunabs ) + .1
   aSDunabs ~ dt( 0 , 0.001 , 2 )
   skappa <- pow(mg,2)/pow(sg,2)
   rkappa <- mg/pow(sg,2)
   mg ~ dunif(0,50)
   sg ~ dunif(0,30)
}
```

Chapter 21

Ordinal Predicted Variable

Contents

> The winner is first, and that's all that he knows, whether
> Won by a mile or won by a nose. But
> Second recalls every inch of that distance, in
> Vivid detail and with haunting persistence.

Very often the predicted variable is ordinal, such as a rating on a scale from 1 to 5: Rate how much you agree with this statement: "Bayesian methods are the most informative, useful, and rational way to analyze data", where 5 = strongly agree, 4 = moderately agree, 3 = neither agree nor disagree, 2 = moderately disagree, 1 = strongly disagree. This sort of response scale is often called a Likert (pronounced LICK-ert) scale (Likert, 1932), and it typically has 5, 7 or more levels (Dawes, 2008). Vast amounts of survey data use Likert rating scales.

We often want to predict the ordinal measure on the basis of other predictors. For example, predictors might include age of respondent, number of years of education, CPU speed of the computer used by respondent, and so forth. To predict ordinal values from metric predictors, we need a way of mapping a weighted linear combination of predictors onto ordinal scale values. The linking function we will use is a cumulative normal function with thresholds. This type of analysis is therefore called ordinal probit regression. It appeared in the bottom row of Table 14.1, p. 312.

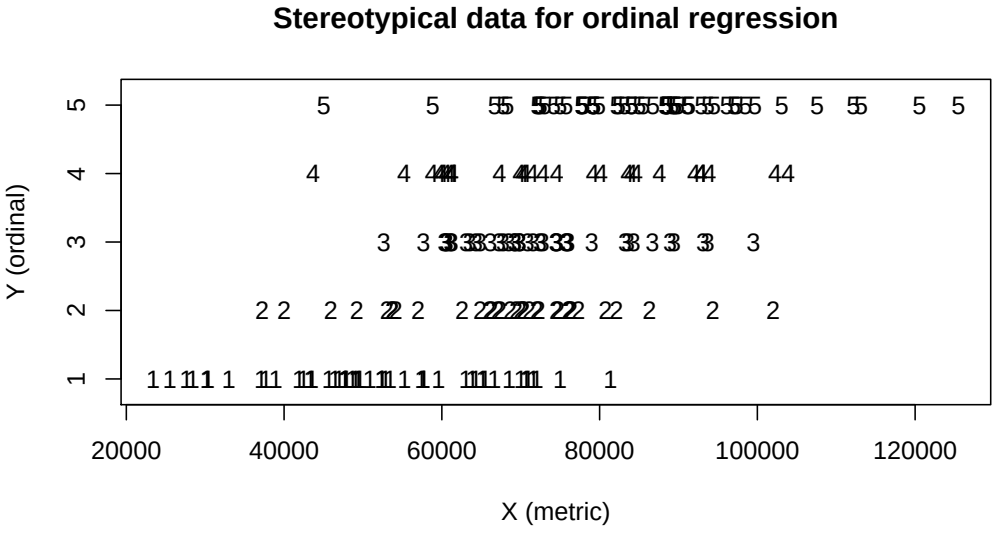

Figure 21.1: Some stereotypical (ctitious) data of the type modeled by ordinal regression. Points are labeled with their y value instead of plotted merely as dots. Although the y values are plotted at equal intervals on the y axis, we really only know their order, not their separation. The conceptual emphasis is on vertical slices through the scatterplot: At any given value of x, what is the probability distribution across the discrete y values?

21.1 Ordinal probit regression

21.1.1 What the data look like

Some stereotypical (ctitious) data that could be modeled by ordinal probit regression are shown in Figure 21.1. The predicted variable, y, is from a 5-point ordinal scale. The predictor variable, x, is metric. If you like concrete examples, imagine that the y value is a rating of overall happiness, and the x value is annual income in dollars. The key aspect of y that makes it ordinal, not metric, is that we don't know whether the distance from 1 to 2 is the same as the distance from 2 to 3, or from 3 to 4, etc. For example, is an increase in happiness from 1 to 2 the same amount of con dential happiness as an increase in happiness from 2 to 3? It may be, but we don't know. All we know is the order of the ratings, not the distances between them.

Regarding Figure 21.1, notice that when x is relatively small, around 40,000, then the y values tend to be 1's or 2's. When x is relatively large, around 100,000, then the y values tend to include more 4's and 5's. But notice also that there is lots of random variability, in the sense that there is a range of y values at any given x value.

21.1.2 The mapping from metric x to ordinal y

Our goal is to construct a model for this sort of data. To accomplish the goal, we start with linear regression on x and then link the continuous prediction to ordinal y values via a thresholded cumulative-normal function. Figure 21.2 shows the underlying mappings from x to y. The right side of the diagram shows simple linear regression. The metric

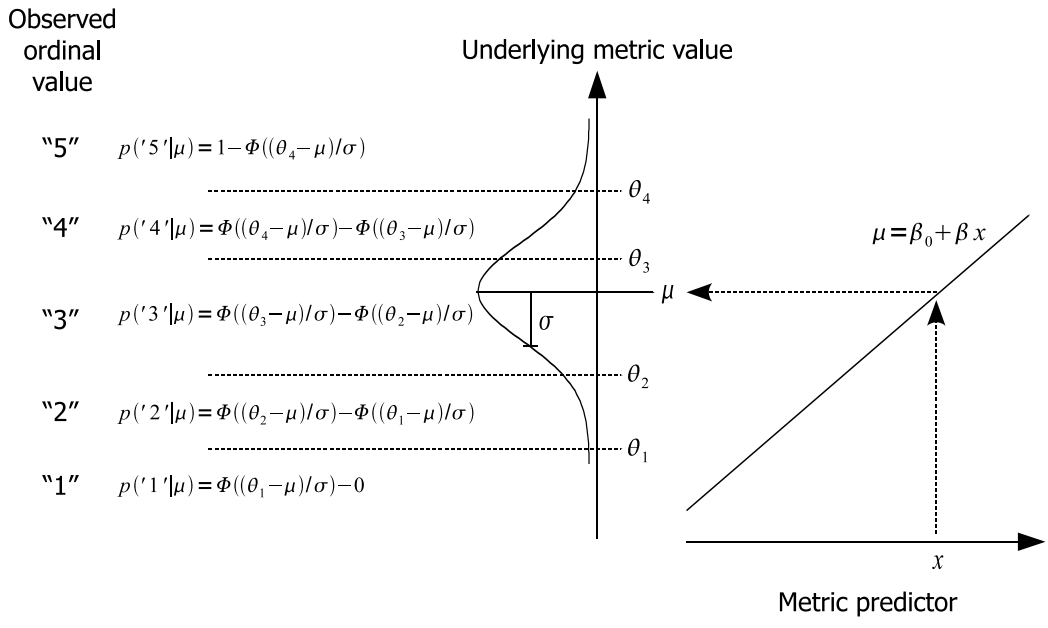

Figure 21.2: The underlying mappings in ordinal probit regression. The lower right shows the predictor value x, which gets mapped to an underlying metric value via a linear relation. There is normally distributed noise in the underlying metric value, which has mean μ and standard deviation σ. The underlying metric value is mapped to an ordinal value depending on which threshold it falls between. The thresholds are not evenly spaced. The probability of falling between thresholds is given by the cumulative normal function Φ.

predictor value, x, gets mapped to an underlying metric tendency μ via the simple linear relationship, $\mu = \beta_0 + \beta x$. The underlying metric value is noisy, distributed around μ as a normal distribution with standard devition σ, which is drawn sideways in the diagram. So far, this is merely simple linear regression.

The new part, on the left of Figure 21.2, gets us from the underlying metric value to an observed ordinal value. To achieve this mapping, the underlying metric scale is carved into intervals by thresholds, located at scale values θ_1, θ_2, and so on, as shown. The threshold values are estimated from the data. When the underlying metric value falls between θ_{k-1} and θ_k, ordinal value k is generated. Because the underlying metric value is normally distributed, the probability that it falls in any interval is given by the cumulative normal distribution.

As you may recall from Section 14.1.7.2 and Figure 14.8, p. 380, the cumulative normal function is denoted Φ. The value of $\Phi(z)$ indicates how much of the normal distribution falls to the left of z, where z is a standardized value. For example, $\Phi(0) = .5$, because half of the standardized normal distribution falls to the left of 0. In R, the cumulative normal function is `pnorm(z)`. In BUGS, the cumulative normal function is `phi(z)`.

To determine how much of the normal distribution falls between two values z_k and z_{k-1}, we simply compute the difference, $\Phi(z_k) - \Phi(z_{k-1})$. We apply this result to the thresholds in Figure 21.2, but first standardize the thresholds relative to the mean and standard deviation of the noise distribution. Thus, the mass of the normal distribution between θ_k and θ_{k-1} is $\Phi((\theta_k - \mu)/\sigma) - \Phi((\theta_{k-1} - \mu)/\sigma)$. This mass is the probability of ordinal value k. In

summary, and as written in Figure 21.2 for each interval,

$$p(\text{“n”}|\theta) = \Phi((\zeta_n - \mu)/\sigma) - \Phi((\zeta_{n-1} - \mu)/\sigma) \quad (21.1)$$

where “n”, in quotes, means that the value n is emitted by the subject. A more traditional, but awkward, way of writing $p(\text{“n”}|\theta)$ is $p(y=n|\theta)$. For the lowest and highest ordinal values, the outside thresholds are at negative and positive infinity. The area under the normal distribution to the left of positive infinity is 1, hence Equation 21.1 becomes $1 - \Phi((\zeta_{n-1} - \mu)/\sigma)$ at the highest ordinal value.

Because the mapping from underlying metric value to ordinal value uses a cumulative normal function, this type of regression might be called “cumulative normal regression” or “phi regression”, by analogy to “logistic regression”. But it's not. Instead, it's called “probit regression” to indicate that a cumulative normal is used as the link function, and it's called “ordinal probit regression” to indicate that the predicted value is ordinal.

21.1.3 The parameters and their priors

Inspection of Figure 21.2 reveals the parameters that are estimated in ordinal probit regression. There are the usual parameters for linear regression, including β_0, β_j for each predictor, and σ. Also, there are the threshold parameters, ζ_1 through ζ_{L-1}, where L is the number of levels in the ordinal scale. It turns out that the intercept trades off with the thresholds, because we can add any constant to the intercept and subtract that constant from the thresholds and have an equivalent model. Therefore we fix the intercept to an arbitrary convenient constant. Also, the noise standard deviation, σ, trades off with the slopes and thresholds, because if we multiple the noise by a constant, we can divide all the slopes and thresholds by that constant and end up with an equivalent model (also translating the thresholds if the intercept is not zero). In summary, we estimate the slopes on the predictors, β_j, and the thresholds, ζ_n, but we fix the intercept β_0 and the noise σ at convenient constants. (Which constants are convenient? This is answered in the description of the R code in Section 21.5.2 (OrdinalProbitRegressionBrugs.R).)

To estimate these parameters, we must define our prior underlying. The prior on the regression coefficients is the same as for linear or logistic regression. For example, each regression coefficient could have a normal prior. If there are several predictors, then a hyperprior could be posited to express the mutual informativeness of the regression coefficients, as in Figure 17.6.

The prior on the thresholds can be (mildly) informed by the scale of the predicted value and by the fact that the thresholds need to be in order, that is, it must be that $\zeta_n > \zeta_{n-1}$. Because the y values are 1 through L, and we will set up the underlying metric value to match that scale, it is reasonable to believe that the thresholds should be approximately 1.5, 2.5, ..., $L - .5$. Our uncertainty in those values is moderate, not extreme, and therefore we may opt to put a normal prior on each threshold with a standard deviation of about 1 unit. This sort of prior, in which each threshold has a normal distribution that is independent of the others, makes it quite possible to have thresholds that are out of order. There are various ways to express prior knowledge that the thresholds must be ordered. One simple way is to put the knowledge into the likelihood function itself, by changing Equation 21.1 to this:

$$p(\text{“n”}|\theta) = \max(0, \Phi((\zeta_n - \mu)/\sigma) - \Phi((\zeta_{n-1} - \mu)/\sigma))$$

Notice that if $_n < _{n\ 1}$, then the probability of "n" is zero. That would be fine if the value n never occurs in the data, but if it does, then threshold values will never be found to be credible.

The prior on the regression coefficients is explained in Exercise 21.2. In that exercise you will also explore the robustness of the posterior to changes in the prior (and you'll find that the posterior is very robust).

21.1.4 Standardizing for MCMC efficiency

To reduce correlation among regression coefficients, and thereby make the MCMC sampling more efficient, the predictor values are standardized before the parameters are estimated. In other words, the x_j values are transformed according to Equation 16.1: $x_j^{(z)} = (x_j - M_j)/SD_j$. For the formulas in this section, the standardized parameters for the standardized predictors all have a superscript, like this $x^{(z)}$, $_j^{(z)}$, and so on. After estimation, the parameters must be transformed back to the original scales of the predictors. To do this back-transformation, consider the following sequence of equalities:

$$p(\text{"}n\text{"} \mid x) = \quad (\ _n^{(z)} \qquad ^{(z)}) = \ ^{(z)} \qquad (\ _{n\ 1}^{(z)} \qquad ^{(z)}) = \ ^{(z)}$$

that's Equation 21.1 with z superscripts

$$p(\text{"}\ n\text{"} \mid x) = \quad (\ _n^{(z)} \qquad ^{(z)}) = \ ^{(z)}$$

notice "$\ n$" on the left-hand side!

$$\begin{aligned}
\text{probit } p(\text{"}\ n\text{"} \mid x) &= (\ _n^{(z)} \qquad ^{(z)}) = \ ^{(z)} \\
&= \frac{1}{^{(z)}} \left(\ _n^{(z)} - \left(_0^{(z)} + \sum_j \ _j^{(z)} x_j^{(z)} \right) \right) \\
&= \frac{1}{^{(z)}} \left(\ _n^{(z)} - \left(_0^{(z)} + \sum_j \ _j^{(z)} \frac{x_j - M_j}{SD_j} \right) \right) \\
&= \frac{1}{^{(z)}} \left(\underbrace{ _0^{(z)} + \sum_j \ _j^{(z)} M_j / SD_j }_{\{z\}_n} - \sum_j \underbrace{ \frac{ _j^{(z)} }{ ^{(z)} SD_j } }_{\{z\}_j} x_j \right) \qquad (21.2) \\
&\quad \text{with } _0 \quad 0 \text{ and } \quad 1
\end{aligned}$$

where all the parameters without z superscripts are on the original scale. Why is $_0$ 0? Because it's arbitrary: we could set it to any value, adding it into the sum on the far right and subtracting it out of the thresholds. Similarly, the value of is arbitrary; we could set it to any value if we multiplied all the thresholds and regression coefficients by that value, and then divided by before passing into the function.

21.1.5 Posterior prediction

The slope and threshold values from Equation 21.2 are arbitrarily determined only relative to $_0$ 0 and 1. Their usefulness is being able to easily compute predicted probabilities

from the original scale values, as follows:

$$p(\text{“}n\text{”} \mid x) = \Phi\left(\theta_n - \sum_j \beta_j x_j\right) - \Phi\left(\theta_{n-1} - \sum_j \beta_j x_j\right) \tag{21.3}$$

This expression (in Equation 21.3) is merely Equation 21.2 re-written with the original-scale parameters set where the underbraces indicate they belong. Exercise 21.3 has you use this equation to make predictions.

21.2 Some examples

Consider the data shown in the top panel of Figure 21.3. There are two metric predictors, labeled X1 and X2 in the graph, and the data to be predicted are ordinal, ranging from 1 to 7. Each data point is labeled with its y value. These data were randomly generated but with very little noise in the y value, so that the transitions from one y level to the next would be very clear. Notice that as X1 increases, the y value tends to decrease. Hence the regression coefficient on X1 should be negative. Notice also that as X2 increases, the y values tends to increase. Hence the regression coefficient on X2 should be positive.

The program in Section 21.5 (OrdinalProbitRegressionBrugs.R), which does Bayesian ordinal probit regression, was run with these data. Aspects of the resulting posterior are shown in the middle and lower portions of Figure 21.3. The first histogram indicates credible values of β_1, which are all negative, as inspection of the trends in the data suggested. The second histogram indicates credible values of β_2, which are all positive, again as inspection of the trends in the data suggested.

The next six histograms show credible values of the thresholds. One thing to notice about the distributions of the thresholds is that they appear to overlap tremendously: The histogram for θ_1 covers much of the same range as the histogram for θ_2, and so on. This overlap would seem to suggest a violation of the necessary ordering of the thresholds, because, it seems, θ_1 could easily be larger than θ_2. What's wrong? Nothing, because credible values of the thresholds are very strongly correlated. The lower part of Figure 21.3 shows pairwise scatterplots of the parameter values, and reveals the extreme correlation. In fact, if you plot histograms of the posterior $\theta_n - \theta_{n-1}$, you will find that the differences do not overlap zero at all; i.e., there is no violation of ordering the thresholds. Exercise 21.1 provides more details regarding how to do this.

The plot of the data, in the top panel of Figure 21.3, also includes examples of level contours from the posterior. A set of six (i.e., $L-1$) level contours comes from a single step in the MCMC chain. The six level contours correspond to the six thresholds, for the β_1 and β_2 values at that step in the chain. The level contours within a set have the same slope because they all correspond to the same β_1 and β_2 values. Different sets of contours have different slopes because they have different β_1 and β_2 values. Because these data were generated with very little noise in the y values, there is little uncertainty in the slopes or thresholds. One indicator of the lack of noise is that fact that between level contours in the plot there is mostly just one value of y.

The level contours in the top panel of Figure 21.3 were derived as follows. Consider the line that marks the transition from the $n-1$ region to the “n” region. The transition occurs when $p(y \geq \text{“}n\text{”} \mid x)$ is 0.5, which means that probit$(p(y \geq \text{“}n\text{”} \mid x)) = 0$. From Equation 21.3

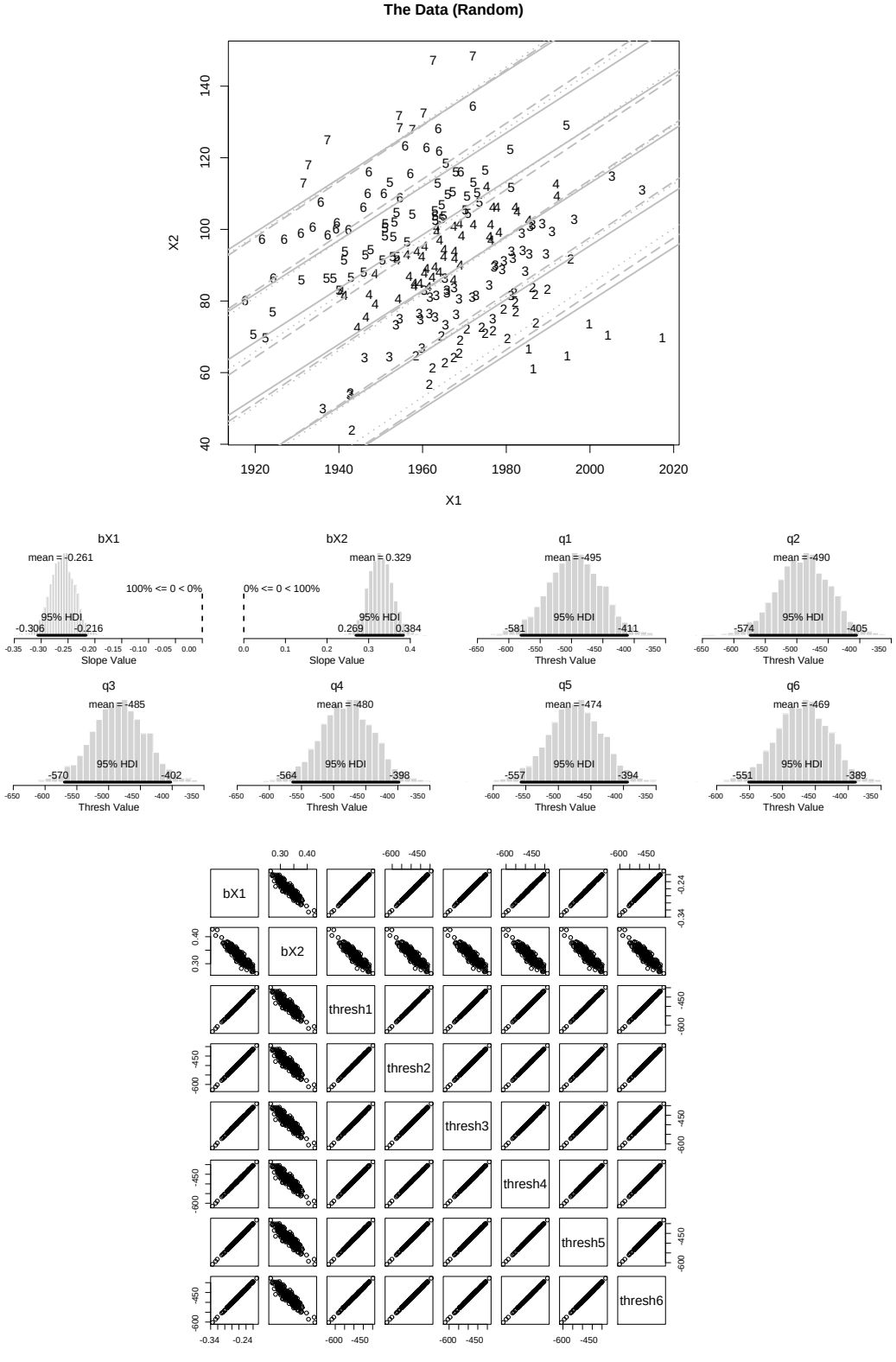

Figure 21.3: Upper panel: Random ordinal data plotted against two predictors, superimposed with a smattering of threshold lines from the posterior. Histograms and lower scatterplots show posterior on the regression coefficients and thresholds.

(and/or Eqn. 21.2), we know

$$\text{probit}\, p(\text{“}n\text{”}|x) = \zeta_n - \sum_j \beta_j x_j$$

It follows that the transition line occurs for x values that satisfy

$$\begin{aligned} 0 &= \zeta_n - \sum_j \beta_j x_j \\ 0 &= \zeta_n - \beta_1 x_1 - \beta_2 x_2 \quad \text{when there are two predictors} \\ x_2 &= \frac{\zeta_n}{\beta_2} + \frac{-\beta_1}{\beta_2} x_1 \end{aligned}$$

The level contours in the top panel of Figure 21.3 used this formula; they are applied in the last few lines of the code in Section 21.5 (OrdinalProbitRegressionBrugs.R).

The previous example used fictitious data so that the concept could be clearly illustrated. Realistic data are typically much messier. Consider, for example, ratings of movies (Moore, 2006), as shown in the top panel of Figure 21.4. The predictor on the abscissa is the year the movie was made, and the predictor on the ordinate is the length, i.e., duration, of the movie in minutes. Notice that ratings are rather noisy: Within any pair of contour lines there occur many different rating values.

Despite the noise in the data, the regression coefficients on the two predictors are very credibly non-zero, as can be seen in the first two histograms in Figure 21.4. Ratings clearly tend to decrease as year of production increases, and ratings clearly tend to increase as length increases. Nevertheless, the degree of uncertainty in the regression coefficients is much larger than the previous example. This uncertainty is apparent in the smattering of posterior level contours superimposed on the data: The slopes of the level contours vary greatly from one set to another.

The posterior distribution of thresholds again shows the strongest correlations, in the lower scatterplots of Figure 21.4. The thresholds never violate the proper order, although exactly where they should be is rather uncertain.

21.2.1 Why are some thresholds outside the data?

In Figure 21.4, the upper panel shows a scatterplot of data with a few of the posterior contour lines superimposed. Notice that some of the extreme threshold contours fall outside all the data points. For example, the lower right of the plot shows that the first and second threshold contours have almost no data below them. How can these thresholds be among the most credible?

The answer lies in gaining a better intuition for the predictions of the ordinal probit model. Figure 21.5 shows the predictions of the model for a single x variable and thresholds set at 1 through 5. The upper panel shows the predictions when the noise is very small compared to the separation between thresholds, namely $\sigma = 0.1$. In this case, the predictions match intuition: When x is between thresholds n and $n + 1$, the most probable response is $n + 1$. It's only when x is very close to a threshold that an adjacent response gains probability.

Moving down Figure 21.5, the successive panels show greater amounts of noise, i.e., larger values of σ. Consider what happens to the predicted response for $x = 2.2$. When σ is small, the highest probability response is 3. As the noise increases, the dominance of this

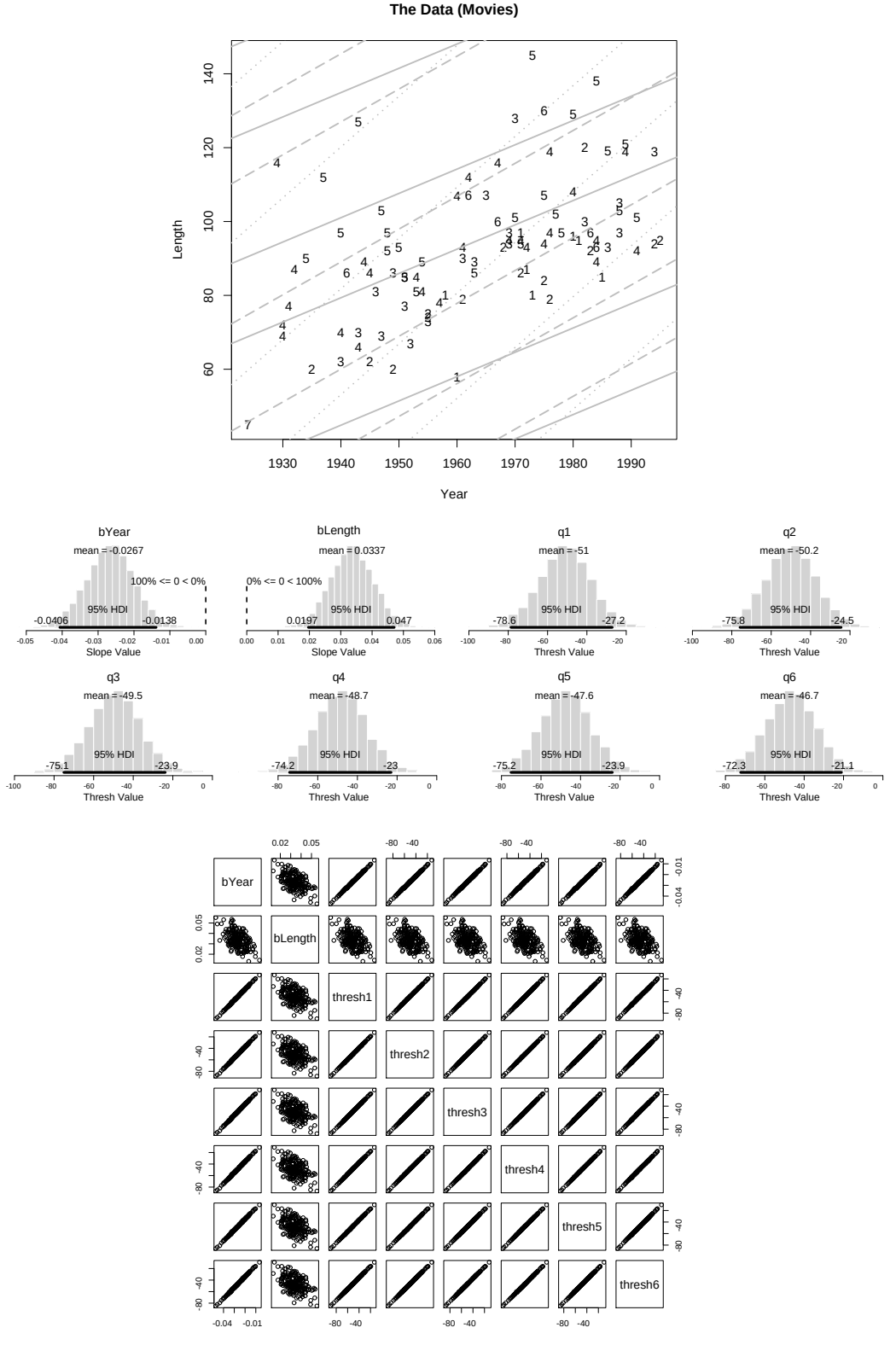

Figure 21.4: Upper panel: Movie ratings from Moore (2006) plotted against Length and Year, superimposed with a smattering of threshold lines from the posterior.. Histograms and lower scatterplots show posterior of the regression coefficients and thresholds.

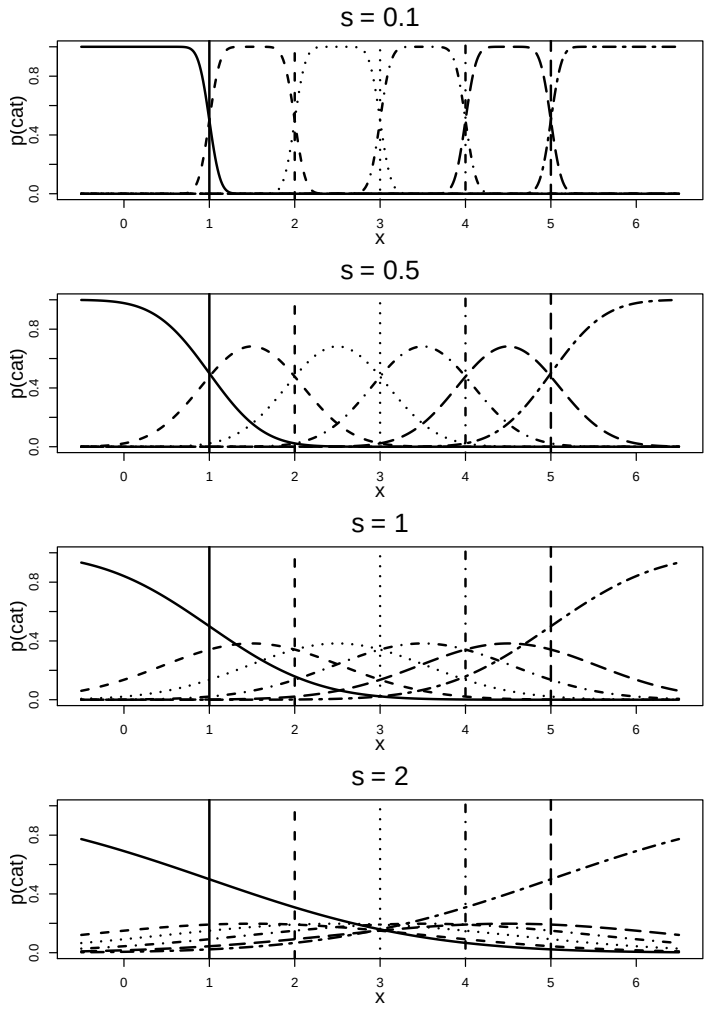

Figure 21.5: Each panel shows the probability of an ordinal response as a function of the stimulus value x. The upper panel is for small, and the lower panel is for large .

response becomes weaker. Finally, in the bottom panel, with high noise, 3 is no longer the dominant response, and instead 1 is! Thus, even though $x = 2.2$, which falls between the 2nd and 3rd thresholds, the highest probability response is 1.

Suppose that the data consist of values scattered between $x = 2$ and $x = 4$. Suppose also that responses are very noisy, with $s = 2$, as in the lowest panel of Figure 21.5. For a point near $x = 2$, the most probable response is 1, even though the point is above the threshold for 1. For a point near $x = 4$, the most probable response is 6, even though the point is far below the threshold for 6. It is this sort of pattern that we see in the movie data of Figure 21.4, in which the responses are very noisy.

21.3 Interaction

Just as in multiple linear regression and multiple logistic regression, there can be multiplicative interaction of the predictors in ordinal probit regression. Much as the logistic function

squashes the predictions, the cumulative normal function squashes the predictions. Hence the prediction surface, for ordinal probit regression with interaction, is much like the prediction surface for logistic regression, as was shown in Fig. 20.4, p. 455. Ordinal probit regression is analogous to logistic regression when there are only two levels of the ordinal variable y, because logistic regression has a predicted variable with two levels. In fact, in ordinal regression, the levels can be groups into "$\leq n$" and "$> n$", and therefore every transition between adjacent levels can be thought of as analogous to logistic regression on two levels. Thus, when there is multiplicative interaction of two predictors, the interaction can express ideas such as conjuction: The y value is "$> n$" when x_1 is large *and* x_2 is large, but not otherwise. A multiplicative interaction can also express an exclusive-OR: The y value is "$> n$" when x_1 is large *or* x_2 is large, but *not* both.

21.4 Relation to linear and logistic regression

This chapter took the core model of multiple linear regression and extended it to ordinal predicted values by mapping the linear prediction through thresholded cumulative normal function. The only real news regarded the mathematical details of dealing with the cumulative normal function, and implementing the model in R and JAGS. Otherwise, the basic concepts of multiple regression apply to ordinal probit regression.

When dealing with ordinal predicted values, some analysts assume, for convenience, that the ordinal scale values are equally spaced on an interval scale. With this assumption, the problem reverts to ordinary linear regression. Often the results of such an analysis will be very similar to ordinal regression, in terms of the regression coefficients on the predictors. But if the emphasis is prediction of the probabilities of each ordinal level, the ordinal regression gives a more direct answer (via Eq. 21.3).

When the predicted variable has only two levels, it is tantamount to an ordinal scale with two levels (because order can be assigned arbitrarily, "1" > "0" or "0" > "1", without changing the results). In this case, ordinal probit regression will yield results essentially equivalent to logistic regression. The only difference will be subtle because of the slight difference in shape between the logistic function and the cumulative normal function. In fact, when there are only two levels of the predicted variable, some analysts prefer probit (cumulative normal) regression to logistic regression. Because the results tend to be so similar, the choice between link function usually is governed by meaningfulness for the domain of application. If the link function is to be thought of as describing normally distributed noise on an underlying linear relation, then the probit is more meaningful. If the analyst finds it meaningful to discuss regression coefficients in terms of log-odds (see Section 20.1.3, p. 452), then the logistic link function is more useful.

21.5 R code

Initialization of the chains uses the MLE from linear regression as a heuristic approximation to start the slope estimates for ordinal regression. The R function `lm()` is used for this purpose. The MLE also returns an estimate of the intercept (which should be close to the mean of the y values, because the x values are standardized), which is used as the arbitrary fixed value for β_0 in the model. The MLE has a residual error, which is used to compute a corresponding standard deviation that is used as the arbitrary fixed value for σ.

(OrdinalProbitRegressionBrugs.R)

```
graphics.off()
rm(list=ls(all=TRUE))
fname = "OrdinalProbitRegressionBrugs"
library(BRugs)            # Kruschke, J. K. (2010). Doing Bayesian dat      a analysis:
                            # A Tutorial with R and BUGS. Academic Press / Elsevier.
#------------------------------------------------- ----------------------------
# THE MODEL.

modelstring = "
# BUGS model specification begins here...
model {
    for( i in 1 : nData ) {
        y[i] ~ dcat( pr[i,1:nYlevels] )
        pr[i,1] <- phi( ( thresh[1] - mu[i] ) / sigma )
        for ( k in 2:(nYlevels-1) ) {
            pr[i,k] <- max( 0 ,   phi( ( thresh[ k ] - mu[i] ) / sigma )
                                  - phi( ( thresh[k-1] - mu[i] ) / sigma ) )
        }
        pr[i,nYlevels] <- 1 - phi( ( thresh[nYlevels-1] - mu[i] ) / si          gma )
        mu[i] <- b0 + inprod( b[1:nPredictors] , x[i,1:nPredictors          ] )
    }
    bPrec <- pow( nYlevels/4 , -2 ) # max plausible slope is 1SD
    for ( j in 1:nPredictors ) {
        b[j] ~ dnorm(0,bPrec) # modest precision because of normali      zed x,y values
    }
    threshPriorPrec <- 1
    for ( k in 1:(nYlevels-1) ) {
        threshPriorMean[k] <- k+0.5
        thresh[k] ~ dnorm( threshPriorMean[k] , threshPriorPrec )
    }
}
# ... end BUGS model specification
" # close quote for modelstring
writeLines(modelstring,con="model.txt")
modelCheck( "model.txt" )

#------------------------------------------------- ----------------------------
# THE DATA.

dataSource = c("Random","Movies")[2]

# The loading of data must produce a matrix called dataMat tha     t has
# one row per datum, where the first column is the ordinal pred      icted value
# and the 2nd - last columns are the predictor values. The colu      mns should
# be named.

if ( dataSource=="Random" ) {
    fname = paste( fname , dataSource , sep="" )
    # Generate some random toy data.
    source( "OrdinalProbitDataGenerator.R" )
    nYlevels = 7
    dataMat = OrdinalProbitDataGenerator( nData = 200 ,
                normPrec=200 , slope=c(-1.0,1.26) , # c(-1.0,1.26) matche        s Movies
                thresh=c(-Inf,seq(-1.2,1.2,length=nYlevels-1),Inf) ,
                nYlevels=nYlevels , makePlots=F , rndSeed=47405 )
    # Change x values to arbitrary non-standardized scales:
    dataMat[,2] =  1963.64 + 18.13 * dataMat[,2]
    dataMat[,3] =  92.87 + 18.26 * dataMat[,3]
```

```
}

if ( dataSource=="Movies" ) {
    fname = paste( fname , dataSource , sep="" )
    dataFram = read.table( "Moore2006data.txt" , header=T )
    rateVals = sort( unique( dataFram[,"Rating"] ) )
    rankVals = match( dataFram[,"Rating"] , rateVals ) # conver t to ranks
    dataMat = cbind( rankVals , dataFram[,"Year"] , dataFram[, "Length"] )
    colnames(dataMat) = c("Rating","Year","Length")
}

# Rename for use by generic processing later:
nData = NROW(dataMat)
x = dataMat[,-1]
predictorNames = colnames(dataMat)[-1]
nPredictors = NCOL(x)
y = as.matrix(dataMat[,1])
predictedName = colnames(dataMat)[1]
nYlevels = max(y)

# Re-center x values at mean, to reduce autocorrelation in MC MC sampling.
# Standardize (divide by SD) to make prior-setting easier.
standardizeCols = function( dataMat ) {
    zDataMat = dataMat
    for ( colIdx in 1:NCOL( dataMat ) ) {
        mCol = mean( dataMat[,colIdx] )
        sdCol = sd( dataMat[,colIdx] )
        zDataMat[,colIdx] = ( dataMat[,colIdx] - mCol ) / sdCol
    }
    return( zDataMat )
}
zx = standardizeCols( x )
# Don't standarize y because they must be integers, 1 to nYlev els

lmInfo = lm( y ~ zx ) # R function returns MLE
b0Init = lmInfo$coef[1]
bInit = lmInfo$coef[-1]
sigmaInit = sqrt(sum(lmInfo$res^2)/nData)

# Get the data into BUGS:
datalist = list(
        x = zx ,
        y = as.vector( y ) , # BUGS does not treat 1-column mat as vector
        nPredictors = nPredictors ,
        nData = nData ,
        nYlevels = nYlevels ,
        sigma = sigmaInit , # fixed, not estimated
        b0 = b0Init  # fixed, not estimated
)
modelData( bugsData( datalist ) )

#------------------------------------------------ ----------------------------
# INTIALIZE THE CHAINS.

nChain = 3
modelCompile( numChains = nChain )

genInitList <- function() {
    list(
```

```
            b = bInit , # from lm(y~zx), above
            thresh = 1:(nYlevels-1)+.5
        )
}
for ( chainIdx in 1 : nChain ) {
    modelInits( bugsInits( genInitList ) )
}

#------------------------------------------------- ----------------------------
# RUN THE CHAINS

# burn in
BurnInSteps = 2000
modelUpdate( BurnInSteps )
# actual samples
samplesSet( c( "b" , "thresh" ) )
stepsPerChain = ceiling(5000/nChain)
thinStep = 20
modelUpdate( stepsPerChain , thin=thinStep )

#------------------------------------------------- ----------------------------
# EXAMINE THE RESULTS

source("plotChains.R")
source("plotPost.R")

checkConvergence = T
if ( checkConvergence ) {
  bSum      = plotChains( "b"      , saveplots=F , filenameroot=fname )
  threshSum = plotChains( "thresh" , saveplots=F , filenamer    oot=fname )
}

# Extract chain values:
zbSamp = NULL
for ( j in 1:nPredictors ) {
    zbSamp = cbind( zbSamp , samplesSample( paste("b[",j,"]",    sep="") ) )
}
chainLength = NROW(zbSamp)
zthreshSamp = NULL
for ( j in 1:(nYlevels-1) ) {
    zthreshSamp = cbind( zthreshSamp ,
                         samplesSample(paste("thresh[",j,"]",sep="")) )
}


# Convert to original scale:
bSamp = zbSamp * matrix( 1/(sigmaInit*apply(x,2,sd)) , byr      ow=TRUE ,
                         ncol=nPredictors , nrow=chainLength )
threshSamp = (1/sigmaInit) * ( zthreshSamp - b0Init +
                 rowSums( zbSamp * matrix( apply(x,2,mean)/apply(x,2,sd)      ,
                                           byrow=TRUE , ncol=nPredictors ,
                                           nrow=chainLength ) ) )
b0 = 0
sigma = 1

# Scatter plots of parameter values, pairwise:
if ( (nPredictors+nYlevels) <= 10 ) { # don't display if too ma        ny
    windows()
    thinIdx = ceiling(seq(1,chainLength,length=200))
```

```
177       pairs( cbind( zbSamp[thinIdx,] , zthreshSamp[thinIdx,] )                ,
178                 labels=c( paste("zb",predictorNames,sep="") ,
179                           paste("zthresh",1:nYlevels,sep="")) )
180       windows()
181       pairs( cbind( bSamp[thinIdx,] , threshSamp[thinIdx,] )   ,
182                 labels=c( paste("b",predictorNames,sep="") ,
183                           paste("thresh",1:nYlevels,sep="")) )
184       dev.copy2eps(file=paste(fname,"PostPairs.eps",sep="          "))
185   }
186
187   # Display the posterior:
188   nPlotPerRow = 5
189   nPlotRow = ceiling((nPredictors+nYlevels-1)/nPlotPerR          ow)
190   nPlotCol = ceiling((nPredictors+nYlevels-1)/nPlotRow)
191   windows(3.5*nPlotCol,2.25*nPlotRow)
192   layout( matrix(1:(nPlotRow*nPlotCol),nrow=nPlotRow,n          col=nPlotCol,byrow=T) )
193   par( mar=c(4,3,2.5,0) , mgp=c(2,0.7,0) )
194   for ( sIdx in 1:nPredictors ) {
195   histInfo = plotPost( bSamp[,sIdx] , xlab="Slope Value" , co          mpVal=0.0 ,
196                           breaks=30 ,
197                           main=bquote( b *.(predictorNames[sIdx]) ) ,
198                           cex.main=1.67 , cex.lab=1.33 )
199   }
200   for ( sIdx in 1:(nYlevels-1) ) {
201   histInfo = plotPost( threshSamp[,sIdx] , xlab="Thresh Val          ue" , compVal=NULL ,
202                           breaks=30 ,
203                           main=bquote( theta * .(sIdx) ) ,
204                           cex.main=1.67 , cex.lab=1.33 )
205   }
206   dev.copy2eps(file=paste(fname,"PostHist.eps",sep=""          ))
207
208   # Plot the data
209   if ( nPredictors == 2 ) {
210   windows()
211   plot( x[,1] , x[,2] , xlab=colnames(x)[1] , ylab=colnames(             x)[2] ,
212           main=paste( "The Data (" , dataSource , ")" , sep="") ,
213           pch=as.character(y) )
214   for ( chainIdx in round(seq(1,chainLength,len=3)) ) {
215     for ( threshIdx in 1:(nYlevels-1) ) {
216       abline( threshSamp[chainIdx,threshIdx]/bSamp[chainId          x,2] ,
217                 -bSamp[chainIdx,1]/bSamp[chainIdx,2] ,
218                 lwd = 2 , lty=chainIdx , col="grey" )
219     }
220   }
221   dev.copy2eps(file=paste(fname,"Data.eps",sep=""))
222
223   } # end if nPredictors == 2
224
225   # Posterior prediction.
226   xProbe = c( 1991 , 94 ) # Note order of values: x1 is year and x2 is          duration.
227   # Set up a matrix for storing the values of p(y|xProbe) at each          step in chain.
228   py = matrix( 0 , nrow=chainLength , ncol=nYlevels )
229   # Step through chain and compute p(y|xProbe) and each step:
230   for ( chainIdx in 1:chainLength ) {
231       yValue = 1
232       py[chainIdx,yValue] = (
233             pnorm( threshSamp[chainIdx,yValue]
234                       - sum( bSamp[chainIdx,] * xProbe ) ) )
235       for ( yValue in 2:(nYlevels-1) ) {
```

```
            py[chainIdx,yValue] = (
                pnorm( threshSamp[chainIdx,yValue]
                        - sum( bSamp[chainIdx,] * xProbe ) )
                 - pnorm( threshSamp[chainIdx,yValue-1]
                            - sum( bSamp[chainIdx,] * xProbe ) ) )
        }
        yValue = nYlevels
        py[chainIdx,yValue] = ( 1 -
            pnorm( threshSamp[chainIdx,yValue-1]
                    - sum( bSamp[chainIdx,] * xProbe ) ) )
}
# Now average across the chain:
pyAve = colMeans( py )
```

21.6 Exercises

Exercise 21.1. [Purpose: Investigate posterior differences of thresholds.] Run the program in Section 21.5 (OrdinalProbitRegressionBrugs.R) on the Movies data (Moore, 2006). At the end of the program, notice that the posterior thresholds are stored in a matrix named threshSamp, one column per threshold.

(A) Make histograms of the differences between every pair of adjacent thresholds, i.e., threshSamp[,n]-threshSamp[,n-1], for n in 2:nYlevels. (Hint: Use plotPost with compVal=0) Are any differences between adjacent thresholds close to zero?

(B) Make a scatterplot of $_4$ against $_3$. (Hint: plot(threshSamp[,4],threshSamp[,3])) Superimpose a line that marks $_4 = _3$. (Hint: abline(0,1)) What is the relation of this scatterplot to the histogram you made in the previous part? In particular, what is the relation of the line in the scatterplot to the histogram?

Exercise 21.2. [Purpose: Examine robustness when prior is changed.] In the program in Section 21.5 (OrdinalProbitRegressionBrugs.R), the priors on the regression coefficients and thresholds are mildly informed. Consider the precision of the normal prior for the regression coefficients. Because the predictors are standardized, their range is approximately 2 to +2. If we have a prior assumption that the steepest possible relationship between x and y maps the lowest value of x to the lowest value of y and the highest value of x to the highest value of y, then the steepest regression line would rise nYlevels as x goes from 2 to +2. Hence the steepest plausible slope is nYlevels/4, and the program makes that value the standard deviation of the prior on each regression coefficient. Consider now the prior on the thresholds. Because the thresholds are separated in the probit unit, and the thresholds should not violate consecutive ordering, it is reasonable to set their standard deviations no larger than 1.

(A) Set the prior on the regression coefficients so that the precision is very small, say 1.0E-6. Run the program on the Movies data. Is the posterior much different?

(B) Set the prior on the thresholds to that the precision is very small, say 1.0E-6. (Set the prior on the regression coefficients back to the original setting.) Run the program on the Movies data. Is the posterior much different?

Exercise 21.3. [Purpose: Predicted values for novel predictor values.] Run the program in Section 21.5 (OrdinalProbitRegressionBrugs.R) on the movies data (Moore, 2006). Consider a movie that was not included in the rated movies. Suppose it was made in 1991, and had a length of 94 minutes. What is the probability of each rating of the movie? In other words,

what is the probability that it would be rated 7, what is the probability that it would be rated 6, and so on. Answer the same question for a movie that has a length of 94 minutes but was made in 1931. Hint: Use Equation 21.3, averaging the result across all the steps in the MCMC chain. Extra special bonus hint: Code for doing this is already included at the end of the program; your job is to understand what it's doing.

Chapter 22

Contingency Table Analysis

Contents

> Count me the hours that we've been together, I'll
> Count you the hours I'm light as a feather, but
> 'Cause every hour you're all that I see, there's
> No telling if there's a contingency.

Consider a situation in which we observe two nominal values for each item measured. For example, suppose there is an election, and we randomly selected people regarding their political party aliation (which is a nominal variable) and their candidate preference (which is also a nominal variable). Presumably the two variables are not independent, which is to say that the proportion of people who prefer candidate A should differ from one political party to another. This is the type of situation addressed in this chapter.

Traditionally, this situation comes under the rubric of chi-square tests of independence. But we will, of course, take a Bayesian approach, and consequently have no need to compute chi-square values (except for comparison with the Bayesian conclusions). There are many advantages of a Bayesian approach. As usual, a significant advantage is never having to compute a p value. Better yet is that the Bayesian analysis provides credible intervals on the conjoint probabilities and on any desired comparison of conditions.

I will call our modeling framework *Poisson exponential ANOVA* because it uses a Poisson likelihood distribution with an exponential link function from an underlying ANOVA model. This terminology is not conventional, and it might even be misleading if readers mistakenly infer from the term "ANOVA" that there is a metric predicted variable involved. Nevertheless, the terminology is highly descriptive of the structural elements of the model.

Table 22.1: Frequencies of different combinations of hair color and eye color. Data from Snee (1974).

Hair Color	Eye Color			
	Blue	Brown	Green	Hazel
Black	20	68	5	15
Blond	94	7	16	10
Brunette	84	119	29	54
Red	17	26	14	14

22.1 Poisson exponential ANOVA

22.1.1 What the data look like

To motivate the model, we need first to understand the structure of the data. An example of the sort of data we'll be dealing with is shown in Table 22.1. The data come from a classroom poll of students at the University of Delaware (Snee, 1974). Respondents reported their hair color and eye color, with each variable split into four nominal levels as indicated in Table 22.1. The cells of the table indicate the frequency with which each combination occurred in the sample. Each respondent falls in one and only one cell of the table.

The data to be predicted are the cell frequencies. The predictors are the nominal variables. This situation is analogous to two-way ANOVA, which also had two nominal predictors, but had several metric values in each cell instead of a frequency.

For data like these, we can ask a number of questions. We could consider about one variable at a time, and ask questions such as, "Are there more brown-eyed people than non-brown-eyed people?", or, "Are there more brunette-haired people than non-brunette haired people?" Those questions are analogous to main effects in ANOVA. But usually we display the conjoint frequencies specifically because we're interested in the relationship between the variables. We would like to know if the distribution of frequencies across one variable depends on, or is contingent upon, the level of the other variable. For example, does the distribution of hair colors depend on eye color, and, specifically, is the proportion of blond-haired people the same for brown-eyed and blue-eyed? These questions are analogous to interaction contrasts in ANOVA.

22.1.2 The exponential link function

The model can be motivated two different ways. One way is simply to start with the two-way ANOVA model and find a way to map the predicted value to frequency data. The predicted value from the ANOVA model could be any value from negative to positive infinity, but frequencies are non-negative. Therefore we must transform the ANOVA predictions to non-negative values, while preserving order. The natural way to do this, mathematically, is via the exponential transformation. But this transformation only gets us to an underlying continuous predicted value, not to the probability of a discrete frequency. A natural candidate for the needed likelihood distribution is the Poisson (described below), which takes a non-negative value and gives a probability for each integer from zero to infinity. But this motivation may seem a bit arbitrary, even if there's nothing wrong with it in principle.

A different motivation was introduced in Section 14.2.1, p. 318. We start by treating the cell frequencies as representative of underlying cell probabilities, and then asking whether

Table 22.2: Example of exponentiated linear model with zero interaction. Margins show the values of the's, and cells show $_{rc} = \exp(\ _0 + \ _r + \ _c)$. Notice that every row has the same relative probabilities, namely, 100, 10, 1. In other words, the row and column attributes are independent.

$_0 = 4.60517$	$_{c1} = 0$	$_{c2} = 2.30259$	$_{c2} = -2.30259$
$_{r1} = 0$	100	1000	10
$_{r2} = 2.30259$	1000	10000	100
$_{r3} = -2.30259$	10	100	1

the two nominal variables are independent of each other. (Recall the definition of independence way back in Section 3.4.3, p. 39.) For example, in Table 22.1, there's a particular marginal probability that hair color is black, and a particular marginal probability that eye color is brown. If hair color and eye color are independent, then the conjoint probability of black hair and brown eyes is the product of the marginal probabilities. The attributes of hair color and eye color are independent if that relationship holds for every cell in the table. Independence of hair color and eye color means that the proportion of black hair among brown-eyed people is the same as the proportion of black hair among blue-eyed people, and so on for all hair and eye colors.

To check for independence of attributes, we need to estimate the marginal probabilities of the attributes. Denote the marginal (i.e., total) frequency of the r^{th} row as $f(r)$, and denote the marginal frequency of the c^{th} column as $f(c)$. Then the estimated marginal probabilities are $f(r)/N$ and $f(c)/N$, where N is the total of the entire table. If the attributes are independent, then the predicted conjoint probability, $\hat{p}(r,c)$, should equal the product of the marginal probabilities, which means $\hat{p}(r,c) = p(r)\ p(c)$, hence $\hat{f}(r,c)/N = f(r)/N\ f(c)/N$. Because the models we've been dealing with involve additive combinations, not multiplicative combinations, we convert the multiplicative expression of independence into an additive expression by using the fact that $\log(ab) = \log(a) + \log(b)$, as follows:

$$\begin{aligned}
\hat{f}(r,c)/N &= f(r)/N\ \ f(c)/N \\
\hat{f}(r,c) &= f(r)\ \ f(c)\ \ 1/N \\
\hat{f}(r,c) &= \exp\big(\underbrace{\log(f(r))}_{\ _r} + \underbrace{\log(f(c))}_{\ _c} + \underbrace{\log(1/N)}_{\ _0}\big) \qquad (22.1)
\end{aligned}$$

If we abstract the form of Equation 22.1 away from the specific frequencies, we get the equation $_{rc} = \exp(\ _r + \ _c + \ _0)$. The idea is that whatever are the values of the's, the resulting 's will obey multiplicative independence. An example is shown in Table 22.2. The choice of 's is shown in the margins of the table, and the resulting's are shown in the cells of the table. Notice that every row has the same relative probabilities, namely, 100, 10, and 1. In other words, the row and column attributes are independent.

We have dealt before with additive combinations of row and column influences, in the context of ANOVA. In ANOVA, $_0$ is a baseline representing the overall central tendency, and $_r$ is a deflection away from baseline due to being in the r^{th} row, and $_c$ is a deflection away from baseline due to being in the c^{th} column. The deflections are constrained to sum to zero, and the example in Table 22.2 respects this constraint.

In ANOVA, when the cell data are not captured by an additive combination of row and column effects, we include an interaction term, denoted here as $_{rc}$. The interaction

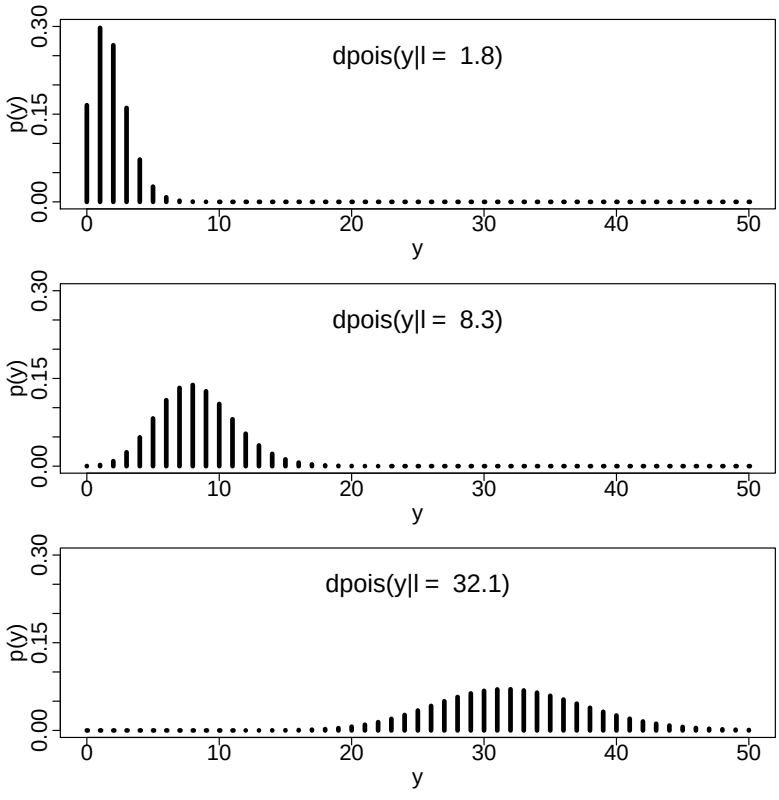

Figure 22.1: Examples of the Poisson distribution. y includes integers from zero to positive infinity.

terms are constrained so that every row and every column sums to zero: $\sum_r \lambda_{rc} = 0 \; \forall c$ and $\sum_c \lambda_{rc} = 0 \; \forall r$. The key idea is that the interaction term in the model, which indicates violations of additivity in standard ANOVA, indicates violations of multiplicative independence in exponentiated ANOVA. To estimate the magnitude of the violation of multiplicative independence, we estimate the magnitude of the interaction coefficients in exponentiation ANOVA. To summarize, the model of the cell tendencies is

$$\lambda_{rc} = \exp(\lambda_0 + \lambda_r + \lambda_c + \lambda_{rc}) \quad \text{with the constraints} \quad \sum_r \lambda_r = 0 \text{ and } \sum_c \lambda_c = 0 \text{ and } \sum_r \lambda_{rc} = 0 \; \forall c \text{ and } \sum_c \lambda_{rc} = 0 \; \forall r \tag{22.2}$$

If the researcher is interested in violations of independence, then the interest is on the magnitudes of the λ_{rc} interaction terms. The model is especially convenient for this purpose, because arbitrary interaction contrasts can be investigated to determine in more detail where the non-independence is arising.

22.1.3 The Poisson likelihood

The value of λ_{rc} in Equation 22.2 is a cell tendency, not a predicted frequency per se. In particular, the value of λ_{rc} can be any non-negative real value, but frequencies can only be integers. What we need to complete the model is a likelihood function that maps the parameter value λ_{rc} to a probability of possible frequencies. The Poisson distribution is a

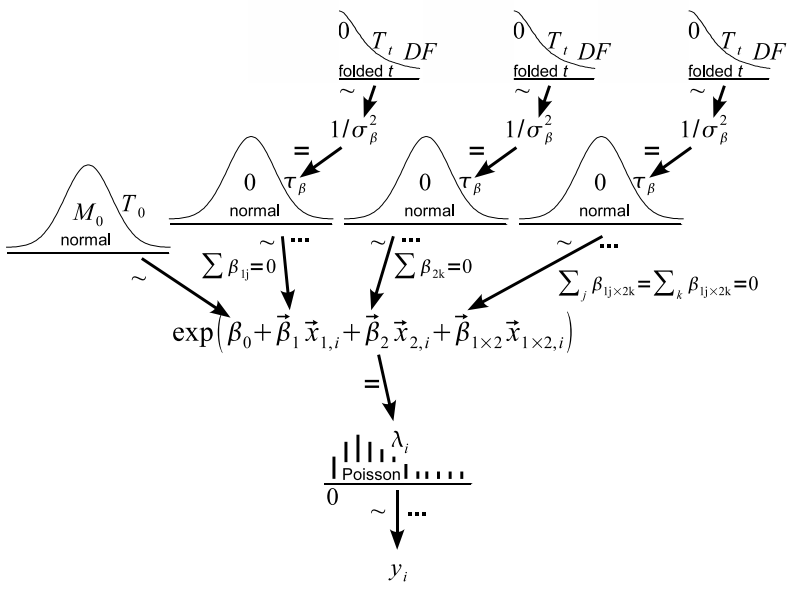

Figure 22.2: Hierarchical dependencies for Poisson-exponential model of a two-way frequency table. Compare with Figure 19.2, p. 424, for two-way ANOVA.

natural choice. The Poisson distribution is named after the French mathematician Simon-Denis Poisson (1781–1840), and is defined as

$$p(y|\lambda) = \frac{\lambda^y \exp(-\lambda)}{y!} \tag{22.3}$$

where y is a non-negative integer and λ is a non-negative real number. The mean of the Poisson distribution is λ. Importantly, the variance of the Poisson distribution is also λ. Thus, in the Poisson distribution, the variance is completely linked to the mean. Examples of the Poisson distribution are shown in Figure 22.1. Notice that the distribution is discrete, having masses only at non-negative integer values. Notice that the visual central tendency of the distribution does indeed correspond with λ. And notice that as λ increases, the width of the distribution also increases. The examples in Figure 22.1 use non-integer vales of λ to emphasize that λ is not necessarily an integer, even though y is an integer.

The Poisson distribution is often used to model discrete occurrences in time (or across space) when the probability of occurrence is the same at any moment in time (e.g., Sadiku & Tofighi, 1999). For example, suppose that customers arrive at a retail store at an average rate of 35 people per hour. Then the Poisson distribution with $\lambda = 35$, is a model of the probability that any particular number of people will arrive in an hour. As another example, suppose that 11.2% of the students at the University of Delaware in the early 1970's had black hair and brown eyes, and suppose that an average of 600 students per term will fill out a survey. That means an average of 67.2 (11.2% 600) students per term will indicate they have black hair and brown eyes. The Poisson distribution $p(y|\lambda=67.2)$, gives the probability that any particular number of people will give that response.

We will use the Poisson distribution as the likelihood function for modeling the probability of the observed frequency, $f(r,c)$, given the mean, λ_{rc}, from Equation 22.2. The idea is that each particular r,c combination has an underlying average rate of occurrence, λ_{rc}. We collect data for a certain period of time, during which we happen to observe particular frequencies, $f(r,c)$, of each combination. From the observed frequencies, we then

Table 22.3: Some toy data for illustrating that interaction contrasts can be more sensitive to interaction than individual cells. (Posterior appears in Figure 22.4.)

	Column Attribute			
Row Attribute	C1	C2	C3	C4
R1	20	20	10	10
R2	20	20	10	10
R3	10	10	20	20
R4	10	10	20	20

underlying average rates.

22.1.4 The parameters and the hierarchical prior

The parameters of the Poisson exponential ANOVA model include all the parameters from standard ANOVA. The prior on those parameters is the same as in standard ANOVA. Figure 22.2 shows the Poisson-exponential model, with its hierarchical prior. Above the linear core, the hierarchy is identical to the two-way ANOVA model in Figure 19.2, p. 424. The diagram retains the original notation from standard ANOVA (e.g., showing $\sum_j \sum_i x_{1,i}$ instead of $\sum_i x_{r,i}$), but hopefully the correspondence of notation is easy to follow.

Below the linear core in Figure 22.2, the diagram simply shows the exponentiation of the ANOVA summation to specify , and the use of the Poisson distribution to specify $p(y|\ )$. This portion of the diagram replaces the normal likelihood of standard ANOVA and the prior on the normal distribution's precision. The Poisson, of course, has no separate precision parameter.

The model is easily implemented in R, BRugs, and BUGS, as listed in Section 22.4 (PoissonExponentialBrugs.R). The code is a straightforward modification of the program for two-factor ANOVA (that was listed in Section 19.3.1 (ANOVAtwowayBRugs.R)). The modifications are described immediately before the listing.

22.2 Examples

Consider again the hair color and eye color data in Table 22.1. When the program in Section 22.4 (PoissonExponentialBrugs.R) is run with these data, the posterior has marginal histograms as shown in Figure 22.3. Inspection of the histograms of the individual cell interactions parameters reveals robustly non-zero interactions. For example, blue eyes and black hair has an interaction parameter that is credibly below zero, which means simply that the combination of blue eyes with black hair happens less frequently than would be expected if eye color and hair color were independent. As another example, brown eyes and black hair has an interaction parameter that is credibly above zero, which means simply that this combination happens more frequently than would be expected if eye color and hair color were independent. Exercise 22.2 has you consider other data and how to interpret main effects and interaction contrasts.

Another way to investigate interaction is by examining specific interaction contrasts. Interaction contrasts can be more sensitive and revealing than single-cell interaction terms. To illustrate this point, consider the toy data in Table 22.3. The upper-left and lower-right cells have higher frequencies than the upper-right and lower-left cells. When a Bayesian

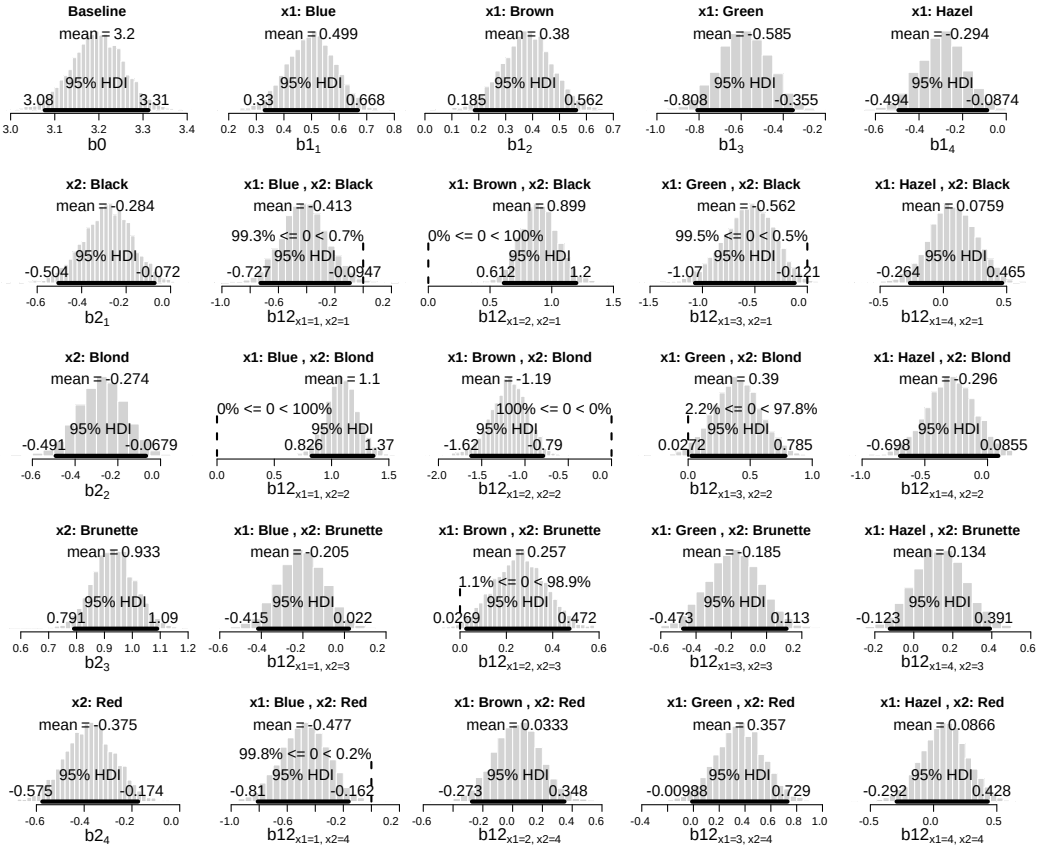

Figure 22.3: Posterior of Poisson exponential model applied to data of Table 22.1.

analysis is conducted on these data, the resulting posterior looks like Figure 22.4. Notice that not one of the single-cell interaction coefficients excludes zero from its HDI. The bottom histogram in Figure 22.4 shows the posterior distribution of an interaction contrast, namely $h[1,1,:,1] - h[1,2,:,1] + h[2,1,:,1] - h[2,2,:,1]$. This contrast takes the average of the interaction coefficients in the top-left and bottom-right cells, and subtracts the average of the interaction coefficients in the top-right and bottom-left cells. The histogram of the contrast magnitude clearly excludes zero from the credible values. Therefore we would conclude that the attributes in Table 22.3 are not independent. Exercise 22.3 has you investigate the pattern of this toy example with different total sample sizes, and compare the Bayesian results with classical chi-square tests.

22.2.1 Credible intervals on cell probabilities

Although the posterior distributions of the beta coefficients are useful for making inferences about differences between cells, the magnitudes of the beta coefficients do not explicitly tell us about the credible cell proportions. To get those cell proportions, we need to use Equation 22.2 to compute the predicted cell mean frequencies and then divide by their total to get the predicted cell proportions. Thus, at every step in the MCMC chain, we compute $m_{rc} = \exp(\beta_0 + \beta_r + \beta_c + \beta_{rc})$, and then we divide, again at each step in the chain, by $\sum_r \sum_c m_{rc}$. The resulting posterior distribution of normalized m_{rc}'s shows the credible proportions of each combination of attributes. Figure 22.5 shows the resulting estimated

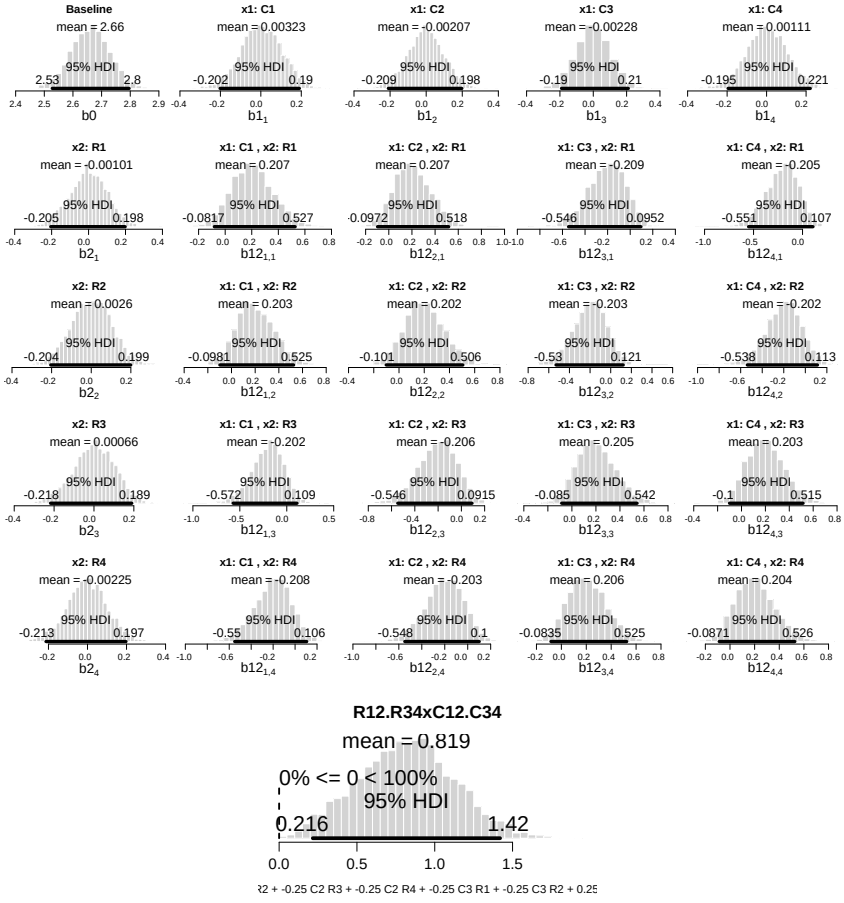

Figure 22.4: Posterior of Poisson exponential model applied to the data of Table 22.3. The interaction contrast excludes zero, but no single interaction coefficient does.

cell probabilities for the hair and eye color data from Table 22.1.

22.3 Log linear models for contingency tables

This chapter only scratches the surface of methods for analyzing count data from nominal predictors. As mentioned earlier, these sorts of data are often displayed as tables, and the counts in each cell are thought of as contingent upon the level of the nominal predictor. Therefore the data are referred to as "contingency tables." There can be more than two predictors, and models are generalized in the same way as ANOVA is generalized to more than two predictors. The formulation presented here has a log linked expressed as $= \exp(_0 + _r + _c + _{rc})$, but this equation can also be written $\log() = _0 + _r + _c + _{rc}$. This latter form lends the usual name for these models: log linear models for contingency tables. This is the terminology to use when you want to explore these models more deeply. Agresti and Hitchcock (2005) provide a brief review of Bayesian log linear models for contingency tables, but the method used in this chapter is not included because the hierarchical ANOVA model was only popularized later (Gelman, 2005, 2006). For a description of Bayesian inference regarding contingency tables with a Poisson ANOVA model, without the Gelman-style hyperprior, see Marin and Robert (2007, pp. 109–118), Congdon (2005, p. 134

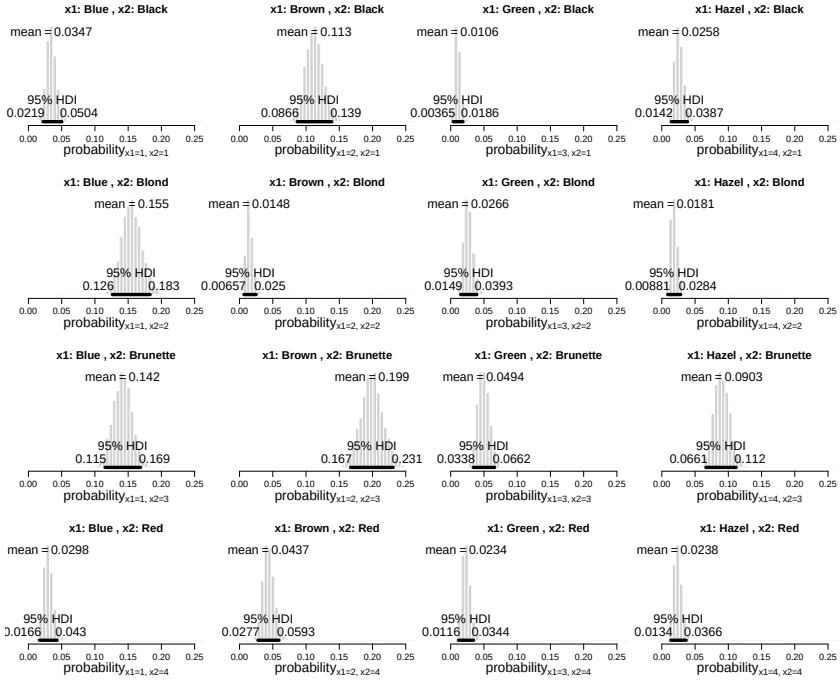

Figure 22.5: Posterior distribution on estimated cell probabilities, for the data of Table 22.1, p. 490. See Section 22.2.1 for details.

and p. 202).

22.4 R code for Poisson exponential model

This program constitutes a small adaptation of the two-way ANOVA program listed in Section 19.3.1 (NOVAtwowayBRugs.R). The only notable changes are that (i) the normal likelihood is changed to a Poisson, (ii) all references to the precision parameter of the normal likelihood are removed, (iii) the y values are not standardized, because they must be non-negative integers, (iv) the initialization of the parameters is based on $\log(y)$, not y, because the parameters are exponentiated to map to λ.

At the end of the program, some code for conducting a chi-square test is included. The purpose of this is merely for comparison of Bayesian results with NHST.

(PoissonExponentialBrugs.R)

```
graphics.off()
rm(list=ls(all=TRUE))
fnroot = "PoissonExponentialBrugs"
library(BRugs)              # Kruschke, J. K. (2010). Doing Bayesian data analysis:
                              # A Tutorial with R and BUGS. Academic Press / Elsevier.
#------------------------------------------------- ----------------------------
# THE MODEL.

modelstring = "
# BUGS model specification begins here...
model {
  for ( i in 1:Ncells ) {
    y[i] ~ dpois( lambda[i] )
```

```
    lambda[i] <- exp( a0 + a1[x1[i]] + a2[x2[i]] + a1a2[x1[i],x2[i]] )
  }
  #
  a0 ~ dnorm(10,1.0E-6)
  #
  for ( j1 in 1:Nx1Lvl ) { a1[j1] ~ dnorm( 0.0 , a1tau ) }
  a1tau <- 1 / pow( a1SD , 2 )
  a1SD <- abs( a1SDunabs ) + .1
  a1SDunabs ~ dt( 0 , 0.001 , 2 )
  #
  for ( j2 in 1:Nx2Lvl ) { a2[j2] ~ dnorm( 0.0 , a2tau ) }
  a2tau <- 1 / pow( a2SD , 2 )
  a2SD <- abs( a2SDunabs ) + .1
  a2SDunabs ~ dt( 0 , 0.001 , 2 )
  #
  for ( j1 in 1:Nx1Lvl ) { for ( j2 in 1:Nx2Lvl ) {
    a1a2[j1,j2] ~ dnorm( 0.0 , a1a2tau )
  } }
  a1a2tau <- 1 / pow( a1a2SD , 2 )
  a1a2SD <- abs( a1a2SDunabs ) + .1
  a1a2SDunabs ~ dt( 0 , 0.001 , 2 )
}
# ... end BUGS model specification
" # close quote for modelstring
# Write model to a file, and send to BUGS:
writeLines(modelstring,con="model.txt")
modelCheck( "model.txt" )

#------------------------------------------------- ----------------------------
# THE DATA.
# Specify data source:
dataSource = c( "HairEye" , "CrimeDrink" , "Toy" )[1]

# Load the data:
if ( dataSource == "HairEye" ) {
  fnroot = paste( fnroot , dataSource , sep="" )
  dataFrame = data.frame( # from Snee (1974)
    Freq = c(68,119,26,7,20,84,17,94,15,54,14,10,5,29,14,16) ,
    Eye  = c("Brown","Brown","Brown","Brown","Blue","Blue","Blue","Blue","Hazel","Hazel","Hazel","H
    Hair = c("Black","Brunette","Red","Blond","Black","Brunette","Red","Blond","Black","Brunette","
  y = as.numeric(dataFrame$Freq)
  x1 = as.numeric(dataFrame$Eye)
  x1names = levels(dataFrame$Eye)
  x2 = as.numeric(dataFrame$Hair)
  x2names = levels(dataFrame$Hair)
  Ncells = length(y)
  Nx1Lvl = length(unique(x1))
  Nx2Lvl = length(unique(x2))
  x1contrastList = list( GREENvHAZEL = c(0,0,1,-1) )
  x2contrastList = list( BLONDvRED = c(0,1,0,-1) )
  x1x2contrastList = list( BLUE.BROWNxBLACK.BLOND
                           = outer(c(-1,1,0,0),c(-1,1,0,0)) )
}

if ( dataSource == "CrimeDrink" ) {
  fnroot = paste( fnroot , dataSource , sep="" )
  dataFrame = data.frame( # from Kendall (1943) via Snee (1974)
    Freq = c(50,88,155,379,18,63,43,62,110,300,14,144) ,
    Crime = c("Arson","Rape","Violence","Theft","Coining","Fraud","Arson","Rape","Violence","Thef
```

```
    Drink = c("Drinker","Drinker","Drinker","Drinker","Dr              inker","Drinker","Nondrink","Nondrink","Nond
  y = as.numeric(dataFrame$Freq)
  x1 = as.numeric(dataFrame$Crime)
  x1names = levels(dataFrame$Crime)
  x2 = as.numeric(dataFrame$Drink)
  x2names = levels(dataFrame$Drink)
  Ncells = length(y)
  Nx1Lvl = length(unique(x1))
  Nx2Lvl = length(unique(x2))
  x1contrastList = list( FRAUDvOTHER = c(-1/5,-1/5,1,-1/5,      -1/5,-1/5) ,
                         FRAUDvRAPE = c(0,0,1,-1,0,0) )
  x2contrastList = list( DRINKERvNON = c(1,-1) )
  x1x2contrastList = list( FRAUD.OTHERxDRINKER.NON
                           = outer(c(-1/5,-1/5,1,-1/5,-1/5,-1/5),c(-1,1)) ,
                           FRAUD.RAPExDRINKER.NON
                           = outer(c(0,0,1,-1,0,0),c(-1,1)) )
}

if ( dataSource == "Toy" ) {
  dataMultiplier = 2 # Try 2 (chi-sq warns) , 6 (p>.05) , 7 (p<.05       ) , 10
  fnroot = paste( fnroot , dataSource , dataMultiplier , sep="        " )
  dataFrame = data.frame(
    Freq = c( 2,2,1,1, 2,2,1,1, 1,1,2,2, 1,1,2,2 ) * dataMultipl          ier ,
    Col  = c("C1","C2","C3","C4", "C1","C2","C3","C4", "C1",        "C2","C3","C4", "C1","C2","C3","C4"),
    Row = c("R1","R1","R1","R1", "R2","R2","R2","R2", "R3",       "R3","R3","R3", "R4","R4","R4","R4") )
  y = as.numeric(dataFrame$Freq)
  x1 = as.numeric(dataFrame$Col)
  x1names = levels(dataFrame$Col)
  x2 = as.numeric(dataFrame$Row)
  x2names = levels(dataFrame$Row)
  Ncells = length(y)
  Nx1Lvl = length(unique(x1))
  Nx2Lvl = length(unique(x2))
  x1contrastList = NULL
  x2contrastList = NULL
  x1x2contrastList = list( R12.R34xC12.C34 = outer(c(1,1,-      1,-1)/2,c(1,1,-1,-1)/2) )
}

# Specify the data in a form that is compatible with BRugs mode    l, as a list:
datalist = list(
  y = y ,
  x1 = x1 ,
  x2 = x2 ,
  Ncells = Ncells ,
  Nx1Lvl = Nx1Lvl ,
  Nx2Lvl = Nx2Lvl
)
# Get the data into BRugs:
modelData( bugsData( datalist ) )

#------------------------------------------------           ----------------------------
# INTIALIZE THE CHAINS.

nchain = 5
modelCompile( numChains = nchain )

if ( F ) {
   modelGenInits() # often won't work for diffuse prior
} else {
```

```
  #  initialization based on data
  theData = data.frame( y=log(y) , x1=factor(x1,labels=x1n          ames) ,
                                    x2=factor(x2,labels=x2names) )
  a0 = mean( theData$y )
  a1 = aggregate( theData$y , list( theData$x1 ) , mean )[,2] - a          0
  a2 = aggregate( theData$y , list( theData$x2 ) , mean )[,2] - a          0
  linpred = as.vector( outer( a1 , a2 , "+" ) + a0 )
  a1a2 = aggregate( theData$y, list(theData$x1,theData$x2       ), mean)[,3] - linpred
  genInitList <- function() {
    return(
      list(
        a0 = a0 ,
        a1 = a1 ,
        a2 = a2 ,
        a1a2 = matrix( a1a2 , nrow=Nx1Lvl , ncol=Nx2Lvl ) ,
        a1SDunabs = sd(a1) ,
        a2SDunabs = sd(a2) ,
        a1a2SDunabs = sd(a1a2)
      )
    )
  }
  for ( chainIdx in 1 : nchain ) {
    modelInits( bugsInits( genInitList ) )
  }
}

#------------------------------------------------                    ----------------------------
# RUN THE CHAINS

# burn in
BurnInSteps = 1000
modelUpdate( BurnInSteps )
# actual samples
samplesSet( c( "a0" ,  "a1" ,  "a2" ,  "a1a2" , "a1SD" , "a2SD" , "a          1a2SD" ) )
stepsPerChain = ceiling(5000/nchain)
thinStep = 500
modelUpdate( stepsPerChain , thin=thinStep )

#------------------------------------------------                    ----------------------------
# EXAMINE THE RESULTS

source("plotChains.R")
source("plotPost.R")

checkConvergence = F
if ( checkConvergence ) {
  sumInfo = plotChains( "a0" , saveplots=F , filenameroot=fn          root )
  sumInfo = plotChains( "a1" , saveplots=F , filenameroot=fn          root )
  sumInfo = plotChains( "a2" , saveplots=F , filenameroot=fn          root )
  sumInfo = plotChains( "a1a2" , saveplots=F , filenameroot=          fnroot )
  readline("Press any key to clear graphics and continue")
  graphics.off()
  sumInfo = plotChains( "a1SD" , saveplots=F , filenameroot=          fnroot )
  sumInfo = plotChains( "a2SD" , saveplots=F , filenameroot=          fnroot )
  sumInfo = plotChains( "a1a2SD" , saveplots=F , filenameroo          t=fnroot )
  readline("Press any key to clear graphics and continue")
  graphics.off()
}
```

```
# Extract and plot the SDs:
a1SDSample = samplesSample("a1SD")
a2SDSample = samplesSample("a2SD")
a1a2SDSample = samplesSample("a1a2SD")
windows(10,3)
layout( matrix(1:3,nrow=1) )
par( mar=c(3,1,2.5,0) , mgp=c(2,0.7,0) )
histInfo = plotPost( a1SDSample , xlab="a1SD" , main="a1 SD " , breaks=30 )
histInfo = plotPost( a2SDSample , xlab="a2SD" , main="a2 SD " , breaks=30 )
histInfo = plotPost( a1a2SDSample , xlab="a1a2SD" , main=" Interaction SD" ,
                     breaks=30 )
dev.copy2eps(file=paste(fnroot,"SD.eps",sep=""))

# Extract a values:
a0Sample = samplesSample( "a0" )
chainLength = length(a0Sample)
a1Sample = array( 0 , dim=c( datalist$Nx1Lvl , chainLength ) )
for ( x1idx in 1:datalist$Nx1Lvl ) {
  a1Sample[x1idx,] = samplesSample( paste("a1[",x1idx,"] ",sep="") )
}
a2Sample = array( 0 , dim=c( datalist$Nx2Lvl , chainLength ) )
for ( x2idx in 1:datalist$Nx2Lvl ) {
  a2Sample[x2idx,] = samplesSample( paste("a2[",x2idx,"] ",sep="") )
}
a1a2Sample = array(0, dim=c( datalist$Nx1Lvl , datalist$N x2Lvl , chainLength ) )
for ( x1idx in 1:datalist$Nx1Lvl ) {
  for ( x2idx in 1:datalist$Nx2Lvl ) {
    a1a2Sample[x1idx,x2idx,] = samplesSample( paste( "a1a2[ ",x1idx,",",x2idx,"]",
                                                     sep="" ) )
  }
}

# Convert to zero-centered b values:
m12Sample = array( 0, dim=c( datalist$Nx1Lvl , datalist$Nx 2Lvl , chainLength ) )
for ( stepIdx in 1:chainLength ) {
  m12Sample[,,stepIdx ] = ( a0Sample[stepIdx]
                            + outer( a1Sample[,stepIdx] ,
                                     a2Sample[,stepIdx] , "+" )
                            + a1a2Sample[,,stepIdx] )
}
b0Sample = apply( m12Sample , 3 , mean )
b1Sample = ( apply( m12Sample , c(1,3) , mean )
             - matrix(rep( b0Sample ,Nx1Lvl),nrow=Nx1Lvl,byrow=T) )
b2Sample = ( apply( m12Sample , c(2,3) , mean )
             - matrix(rep( b0Sample ,Nx2Lvl),nrow=Nx2Lvl,byrow=T) )
linpredSample = array(0,dim=c(datalist$Nx1Lvl,datalis t$Nx2Lvl,chainLength))
for ( stepIdx in 1:chainLength ) {
  linpredSample[,,stepIdx ] = ( b0Sample[stepIdx]
                                + outer( b1Sample[,stepIdx] ,
                                         b2Sample[,stepIdx] , "+" ) )
}
b1b2Sample = m12Sample - linpredSample

# Plot b values:
windows((datalist$Nx1Lvl+1)*2.75,(datalist$Nx2Lvl+1 )*2.25)
layoutMat = matrix( 0 , nrow=(datalist$Nx2Lvl+1) , ncol=(d atalist$Nx1Lvl+1) )
layoutMat[1,1] = 1
layoutMat[1,2:(datalist$Nx1Lvl+1)] = 1:datalist$Nx1Lv l + 1
layoutMat[2:(datalist$Nx2Lvl+1),1] = 1:datalist$Nx2Lv l + (datalist$Nx1Lvl + 1)
```

```
250 layoutMat[2:(datalist$Nx2Lvl+1),2:(datalist$Nx1Lvl+          1)] = matrix(
251       1:(datalist$Nx1Lvl*datalist$Nx2Lvl) + (datalist$Nx2Lv          l+datalist$Nx1Lvl+1) ,
252       ncol=datalist$Nx1Lvl , byrow=T )
253 layout( layoutMat )
254 par( mar=c(4,0.5,2.5,0.5) , mgp=c(2,0.7,0) )
255 histinfo = plotPost( b0Sample , xlab=expression(beta * 0) ,          main="Baseline" ,
256                              breaks=30   )
257 for ( x1idx in 1:datalist$Nx1Lvl ) {
258   histinfo = plotPost( b1Sample[x1idx,] , xlab=bquote(beta          *1[.(x1idx)]) ,
259                                main=paste("x1:",x1names[x1idx]) , breaks=30 )
260 }
261 for ( x2idx in 1:datalist$Nx2Lvl ) {
262   histinfo = plotPost( b2Sample[x2idx,] , xlab=bquote(beta          *2[.(x2idx)]) ,
263                                main=paste("x2:",x2names[x2idx]) , breaks=30 )
264 }
265 for ( x2idx in 1:datalist$Nx2Lvl ) {
266   for ( x1idx in 1:datalist$Nx1Lvl ) {
267     hdiLim = HDIofMCMC(b1b2Sample[x1idx,x2idx,])
268     if ( hdiLim[1]>0 | hdiLim[2]<0 ) { compVal=0 } else { compVal=        NULL }
269     histinfo = plotPost( b1b2Sample[x1idx,x2idx,] , breaks=3          0 , compVal=compVal ,
270                  xlab=bquote(beta*12[list(x1==.(x1idx),x2==.(x2idx))          ]) ,
271                  main=paste("x1:",x1names[x1idx],",  x2:",x2names[x2id          x])   )
272   }
273 }
274 dev.copy2eps(file=paste(fnroot,"b.eps",sep=""))
275
276 # Display contrast analyses
277 nContrasts = length( x1contrastList )
278 if ( nContrasts > 0 ) {
279     nPlotPerRow = 5
280     nPlotRow = ceiling(nContrasts/nPlotPerRow)
281     nPlotCol = ceiling(nContrasts/nPlotRow)
282     windows(3.75*nPlotCol,2.5*nPlotRow)
283     layout( matrix(1:(nPlotRow*nPlotCol),nrow=nPlotRow,n          col=nPlotCol,byrow=T) )
284     par( mar=c(4,0.5,2.5,0.5) , mgp=c(2,0.7,0) )
285     for ( cIdx in 1:nContrasts ) {
286         contrast = matrix( x1contrastList[[cIdx]],nrow=1) # make          it a row matrix
287         incIdx = contrast!=0
288         histInfo = plotPost( contrast %*% b1Sample , compVal=0 , bre      aks=30 ,
289                    xlab=paste( round(contrast[incIdx],2) , x1names[incIdx          ] ,
290                                c(rep("+",sum(incIdx)-1),"") , collapse=" " ) ,
291                    cex.lab = 1.0 ,
292                    main=paste( "X1 Contrast:", names(x1contrastList)[cIdx          ] ) )
293     }
294     dev.copy2eps(file=paste(fnroot,"x1Contrasts.eps",se          p=""))
295 }
296 #
297 nContrasts = length( x2contrastList )
298 if ( nContrasts > 0 ) {
299     nPlotPerRow = 5
300     nPlotRow = ceiling(nContrasts/nPlotPerRow)
301     nPlotCol = ceiling(nContrasts/nPlotRow)
302     windows(3.75*nPlotCol,2.5*nPlotRow)
303     layout( matrix(1:(nPlotRow*nPlotCol),nrow=nPlotRow,n          col=nPlotCol,byrow=T) )
304     par( mar=c(4,0.5,2.5,0.5) , mgp=c(2,0.7,0) )
305     for ( cIdx in 1:nContrasts ) {
306         contrast = matrix( x2contrastList[[cIdx]],nrow=1) # make          it a row matrix
307         incIdx = contrast!=0
308         histInfo = plotPost( contrast %*% b2Sample , compVal=0 , bre      aks=30 ,
```

```
                    xlab=paste( round(contrast[incIdx],2) , x2names[incIdx ] ,
                                c(rep("+",sum(incIdx)-1),"") , collapse=" " ) ,
                    cex.lab = 1.0 ,
                    main=paste( "X2 Contrast:", names(x2contrastList)[cIdx ] ) )
    }
    dev.copy2eps(file=paste(fnroot,"x2Contrasts.eps",se p=""))
}
#
nContrasts = length( x1x2contrastList )
if ( nContrasts > 0 ) {
    nPlotPerRow = 5
    nPlotRow = ceiling(nContrasts/nPlotPerRow)
    nPlotCol = ceiling(nContrasts/nPlotRow)
    windows(3.75*nPlotCol,2.5*nPlotRow)
    layout( matrix(1:(nPlotRow*nPlotCol),nrow=nPlotRow,n col=nPlotCol,byrow=T) )
    par( mar=c(4,0.5,2.5,0.5) , mgp=c(2,0.7,0) )
    for ( cIdx in 1:nContrasts ) {
        contrast = x1x2contrastList[[cIdx]]
        contrastArr = array( rep(contrast,chainLength) ,
                             dim=c(NROW(contrast),NCOL(contrast),chainLength) )
        contrastLab = ""
        for ( x1idx in 1:Nx1Lvl ) {
          for ( x2idx in 1:Nx2Lvl ) {
            if ( contrast[x1idx,x2idx] != 0 ) {
              contrastLab = paste( contrastLab , "+" ,
                                   signif(contrast[x1idx,x2idx],2) ,
                                   x1names[x1idx] , x2names[x2idx] )
            }
          }
        }
        histInfo = plotPost( apply( contrastArr * b1b2Sample , 3 , su m ) ,
                    compVal=0 , breaks=30 , xlab=contrastLab , cex.lab = 0.75 ,
                    main=paste( names(x1x2contrastList)[cIdx] ) )
    }
    dev.copy2eps(file=paste(fnroot,"x1x2Contrasts.eps", sep=""))
}

# Compute credible cell probability at each step in the MCMC c hain
lambda12Sample = 0 * b1b2Sample
for ( chainIdx in 1:chainLength ) {
    lambda12Sample[,,chainIdx] = exp(
        b0Sample[chainIdx]
        + outer( b1Sample[,chainIdx] , b2Sample[,chainIdx] , "+" )
        + b1b2Sample[,,chainIdx] )
}
cellp = 0 * lambda12Sample
for ( chainIdx in 1:chainLength ) {
    cellp[,,chainIdx] = ( lambda12Sample[,,chainIdx]
                          / sum( lambda12Sample[,,chainIdx] ) )
}
# Display credible cell probabilities
windows((datalist$Nx1Lvl)*2.75,(datalist$Nx2Lvl)*2. 25)
layoutMat = matrix( 1:(datalist$Nx2Lvl*datalist$Nx1Lvl ) ,
                    nrow=(datalist$Nx2Lvl) , ncol=(datalist$Nx1Lvl) , byrow =T )
layout( layoutMat )
par( mar=c(4,1.5,2.5,0.5) , mgp=c(2,0.7,0) )
maxp = max( cellp )
for ( x2idx in 1:datalist$Nx2Lvl ) {
  for ( x1idx in 1:datalist$Nx1Lvl ) {
```

```
368       histinfo = plotPost( cellp[x1idx,x2idx,] ,
369                   breaks=seq(0,maxp,length=51) , xlim=c(0,maxp) ,
370                   xlab=bquote(probability[list(x1==.(x1idx),x2==.(x2i              dx))]) ,
371                   main=paste("x1:",x1names[x1idx],",  x2:",x2names[x2id          x]) ,
372                   HDItextPlace=0.95 )
373     }
374   }
375   dev.copy2eps(file=paste(fnroot,"CellP.eps",sep=""))
376
377
378   #=============================================================================
379   # Conduct NHST Pearson chi-square test of independence.
380
381   # Convert dataFrame to frequency table:
382   obsFreq = matrix( 0 , nrow=Nx1Lvl , ncol=Nx2Lvl )
383   for ( x1idx in 1:Nx1Lvl ) {
384     for ( x2idx in 1:Nx2Lvl ) {
385       obsFreq[x1idx,x2idx] = y[ dataFrame[,2]==x1names[x1idx       ]
386                                 & dataFrame[,3]==x2names[x2idx] ]
387     }
388   }
389   obsFreq = t(obsFreq) # merely to match orientation of histog       ram display
390   chisqtest = chisq.test( obsFreq )
391   print( "obs :" )
392   print( chisqtest$observed )
393   print( "( obs - exp )^2 / exp :" )
394   print( ( chisqtest$observed - chisqtest$expected )^2 / chi          sqtest$expected )
395   print( chisqtest )
396
397   #=============================================================================
```

22.5 Exercises

Exercise 22.1. [Purpose: Sample size and precision of estimate.] Consider the data regarding hair color and eye color in Table 22.1, which is available for use in the code of Section 22.4 (PoissonExponentialBrugs.R).

(A) Divide the original data frequencies by 2 and take the ceiling() of the result. Thus, the modified data are ceiling(c(68,...,16) / 2) . This modification (nearly) preserves the relative proportions in each cell but halves the sample size. Run the program with these modified data, show the histograms, and compute the width of the 95% HDI's for the contrasts.

(B) Multiply the original data vector by 5, so that all the cell frequencies are five times larger than the original. This modification preserves the relative proportions in each cell but quintuples the sample size. Run the program again. Show the histograms. Compute the width of the 95% HDI's for the contrasts. How do the precision of the contrasts differ for the reduced and the enlarged data?

Exercise 22.2. [Purpose: Explore "main effects" and interactions in a contingency table.] The data section of the program in Section 22.4 (PoissonExponentialBrugs.R) includes data regarding the type of crime committed by convicted criminals and whether or not the criminal is a regular drinker of alcoholic beverages (Snee, 1974).

(A) Run the program with these data selected. Is there a credible dependence of the attributes? Describe the interaction in terms of the cell levels of the attributes (i.e.,

type of crime and drinker or non-drinker).

(B) Is there a ("simple") main effect of fraud versus rape? That is, marginalizing across drinking, is there a credible difference in the proportion of criminals who committed fraud or committed rape? Show the results of an appropriate contrast.

(C) Is the difference from the previous part the same among drinkers and non-drinkers? In other words, is the effect of criminal type the same at all levels of drinking? Show the results of an appropriate interaction contrast.

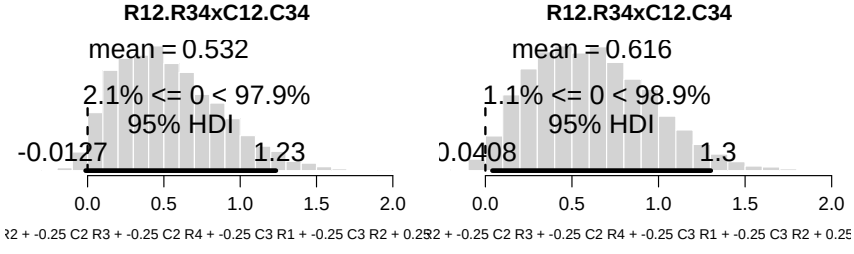

Figure 22.6: For Exercise 22.3. Left histogram shows posterior interaction contrast for the "toy" data when dataMultiplier =6. A chi-square test (not shown) indicates that $p > .05$. Right histogram shows posterior interaction contrast for the "toy" data when dataMultiplier =7. A chi-square test (not shown) indicates that $p < .05$.

Exercise 22.3. [Purpose: Compare Bayesian analysis with chi-square test for independence.] This exercise assumes that you have some familiarity with traditional chi-square tests of independence. In the code of Section 22.2.4 (BernPoissonExponentialBrugs.R), the data section includes some "toy" data, which we can manipulate and compare results of Bayesian and chi-square analyses.

(A) Set the dataMultiplier to 6, and run the program. Include the histograms of the parameters and of the interaction contrast. Does the 95% HDI of the contrast include zero? The end of the program runs a chi-square test. What is the p value from the test? Hint: See Figure 22.6.

(B) Set the dataMultiplier to 7, and run the program. Include the histograms of the parameters and of the interaction contrast. Does the 95% HDI of the contrast include zero? The end of the program runs a chi-square test. What is the p value from the test? Hint: See Figure 22.6.

(C) . Typical chi-square tests rely on approximating the sampling distribution of discrete Pearson chi-square values (i.e., $\sum_r \sum_c (f_{rc} - \hat{f}_{rc})^2/\hat{f}_{rc}$) with the continuous chi-square distribution (which derives from the sum of standardized normal values). When the expected frequencies, $\hat{f}_{rc}$, are too small, then the approximation is not very good, and the estimated p value may be wrong. A usual heuristic for declaring the chi-square test to be suspect is when (at least 10% of the) expected frequencies, $\hat{f}_{rc}$, are less than 5. This is why computer packages will issue warnings when expected values are too small. Bayesian analysis has no such problems. Set the dataMultiplier to 2, and run the program. Does the Bayesian analysis complain or do anything wrong? (No.) The end of the program runs a chi-square test. Is there a warning message? (Yes; report what it is.)

Chapter 23

Tools in the Trunk

Contents

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

This chapter includes a few a few important topics that can apply to many different models throughout the book. The first topic is how to report a Bayesian analysis in a scientific journal. The second topic is the details behind computing an HDI.

The third issue is something that has not come up explicitly very often in the book, but lurks in the shadows all the time. This is the topic of reparameterization. For example, the beta distribution has shape parameters a and b, but we regularly transformed them into parameters ω and κ. The complication of reparameterization is that a prior distribution on a parameter is not the same on the transformed parameter. In particular, if we specify a prior on a parameter, but then transform the parameter, then the prior on the transformed parameter is different. (The quatrain at the beginning of the chapter has reparameterization in mind.)

23.1 Reporting a Bayesian analysis

Bayesian data analyses are not yet standard procedure in many fields of research, and no conventional format for reporting them has been established. Therefore the researcher who reports a Bayesian analysis must be sensitive to the background knowledge of his or her specific audience, and frame the description accordingly. Once I was assigned an exercise to students in which they had to write up the results of a Bayesian analysis as it would appear in a research journal. Because I am a psychologist, a student earnestly asked me, "Do we have to write it as if it were for a psychology journal or for a science journal?" After swallowing my feeling of injury at the implication that psychology is not science, I politely asked the student to clarify the question. The student said that for a psychology journal you have to explain what you did, but for a science journal the reader has to puzzle it out."

23.1.1 Essential points

When reporting a Bayesian analysis, the writer must address the following essential points:

- **Motivate the use of Bayesian (non-NHST) analysis.** Many audiences, including journal editors and reviewers, are used to NHST and unfamiliar with Bayesian methods. You may motivate your use of Bayesian data analysis on several grounds, depending in part on the particular application. For example, Bayesian models are designed to be appropriate to the data structure, without having to make approximation assumptions typical in NHST. The inferences from a Bayesian analysis are richer and more informative than NHST. And, of course, there is no reliance on *p* values.

- **Clearly describe the model and its parameters.** Because the posterior distribution is a distribution over parameter values, the parameters must be clearly defined. This task of describing the model can be arduous for complex hierarchical models, but it is necessary and crucial if your analysis is to mean anything to your audience. Because the model refers to the data and their structure, the model description requires that the structure of the data has already been explained.

- **Clearly describe and justify the prior.** It is crucial to convince your audience that your prior is appropriate and does not predetermine the outcome. There is no escape from this requirement, even when using "objective" or "uninformed" priors, because there is no unique choice of such priors and even they can have major consequences for model comparison. The prior should be amenable to a skeptical audience. The prior should be at least mildly informed to match the scale of the data being modeled. If there is copious previous research using very similar models, it should not be ignored. Optionally, as mentioned again below, it may be useful to try different priors and report the robustness of the posterior.

- **Mention the MCMC details, especially evidence that the chains were converged** (plenty of burn-in and no orphaned chains) and not clumpy (low auto-correlation from sufficient thinning). Also state how many points there are in the final MCMC sample. Usually this section of the report can be brief, if your audience believes that you know what you're doing and therefore took adequate caution to generate a trustworthy and representative sample from the posterior. See Section 23.2 for a reminder of how to do this.

Interpret the posterior. Many models have dozens or even hundreds of parameters, and therefore it is impossible to summarize all of them. The choice of which parameters or contrasts to report is driven by domain-specific theory and by the results themselves. You want to report the parameters and contrasts that are theoretically meaningful, and those that showed effects driven by data, whether expected or not. Reporting of HDI's can be done in text alone, to save space (though histograms of posteriors are useful for explanation in a textbook, may be necessary in a concise report.) Include scatterplots of correlated important parameters if the scatter is not bivariate Gaussian. Be sure to describe effects of shrinkage if appropriate. If your model includes interactions of predictors, be careful how you interpret lower-order effects. Finally, if you are using a ROPE for posterior interpretation, justify its limits.

23.1.2 Optional points

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

Robustness of the posterior for different priors. When there is real or imagined contention about the prior, it can be most convincing simply to conduct the analysis with different priors and demonstrate that the essential conclusions from the posterior do not change. Which priors should be used? Those that are believed by your audience, such as reviewers and editors of the journal to which the report is submitted. This uncertainty in the choice of priors may seem unappealing, but in fact it accurately reflects the way science gets done, by incorporating previously conducted research and addressing currently active researchers.

Posterior predictive check. By generating simulated data from the credible parameter values of the model, and examining the qualities of the simulated data, the veracity of the model can be further bolstered, if the simulated data resemble the actual data. On the other hand, if the simulated data are discrepant from the actual data in systematic and interpretable ways, then the posterior predictive check can inspire new research and new models.

Power analysis. For example, if there is only a weak effect in your results, what sample size would be needed to achieve some desired precision in the estimate? If you found a credible difference, what was the retrospective power of your experiment and what is its replication power? This sort of information can be useful not only for the researcher's own planning, but it can also be useful to the audience of the report to anticipate potential follow-up research and to assess the robustness of the currently reported results.

23.1.3 Helpful points

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

Post the raw data. There are two benefits of posting the original data. One benefit is that subsequent researchers can analyze the data with different models. New insights can be gained by alternative modeling interpretations. The longevity of the original

research is enhanced. A second bene t is that if an exact or near-exact replication is conducted, the original data set can be concatenated with the new data set, to enhance the sensitivity of the new data.

Post the MCMC sample of the posterior. There are two bene ts of making the posterior publicly available. One is that other researchers can explore the posterior for e ects and comparisons that were not in the report. Complex models have many parameters, and no single report can cover every possible perspective on the posterior distribution. The longevity and impact of the research is thereby enhanced. A second bene t is that if subsequent researchers do follow-up research with a similar design and model, then the posted posterior can inform the prior of subsequent analysis. Because the full posterior automatically incorporates all the covariations between all the parameters, the full posterior can be more useful than summaries of marginal distributions in a report.

23.2 MCMC burn-in and thinning

When using a Markov chain to generate a Monte Carlo sample from a distribution, we want to be sure that the resulting chain is a truly representative sample from the distribution. There are several ways in which the chain might fail to be representative.

One problem is that the chains might start far from the true mode(s) of the distribution, and therefore the early steps in the chain might be unrepresentative of the distribution. The main way to mitigate this problem is to start the chains intelligently, instead of randomly, if you can. If you know a point that is likely to be in the midst of the distribution, try starting the chains there. Many of the programs in this book use this mode. For example, in the multiple linear regression program of Section 17.5.1 (MultipleLinearRegressionBrugs.R), the chains were started at the maximum likelihood estimate, because it was assumed that the posterior will be dominated by the data, not by the prior. Although a good starting point reduces the time it takes for the chains to nd the bulk of the distribution, the early steps are still not necessarily representative because we still need to let the chains run a while to dilute the in uence of the initial value; see Figure 7.2, p. 121. This initial set of steps is called the burn-in period.

Another problem is that the chains might not change value much from step to step, and therefore take a long time to generate values from the entire breadth of the distribution. When consecutive steps have similar values, the chain is said to be highly autocorrelated. The problem is not merely taking a long time to generate the range of the distribution. The main problem is that an autocorrelated chain is "clumpy" and over-represents some values and under-represents other values. To primary way to mitigate this problem is to thin the chain: Instead of using every step in the chain, we only use every m^{th} step, where m could be 50 or 500 or larger, depending on the model and data.

Autocorrelation is measured simply as the correlation of the chain values with the values L steps behind (or ahead), where L is called the "lag". Expressed in terms of R code, if the chain is the vector v and it has W components, then the autocorrelation at Lag L is ACF(L) = `cor( v[1:(W-L)] , v[(L+1):W] )`. At $L=0$, the ACF(0) is 1.0, of course. If consecutive steps in the chain are very similar, then ACF(1) will be close to 1.0. But if consecutive steps in the chain are uncorrelated, then ACF(1) will be close to zero.

Another problem is that it is possible for chains to get stuck in unrepresentative regions of parameter space if they are initialized poorly or take a bad jump. As a check that the

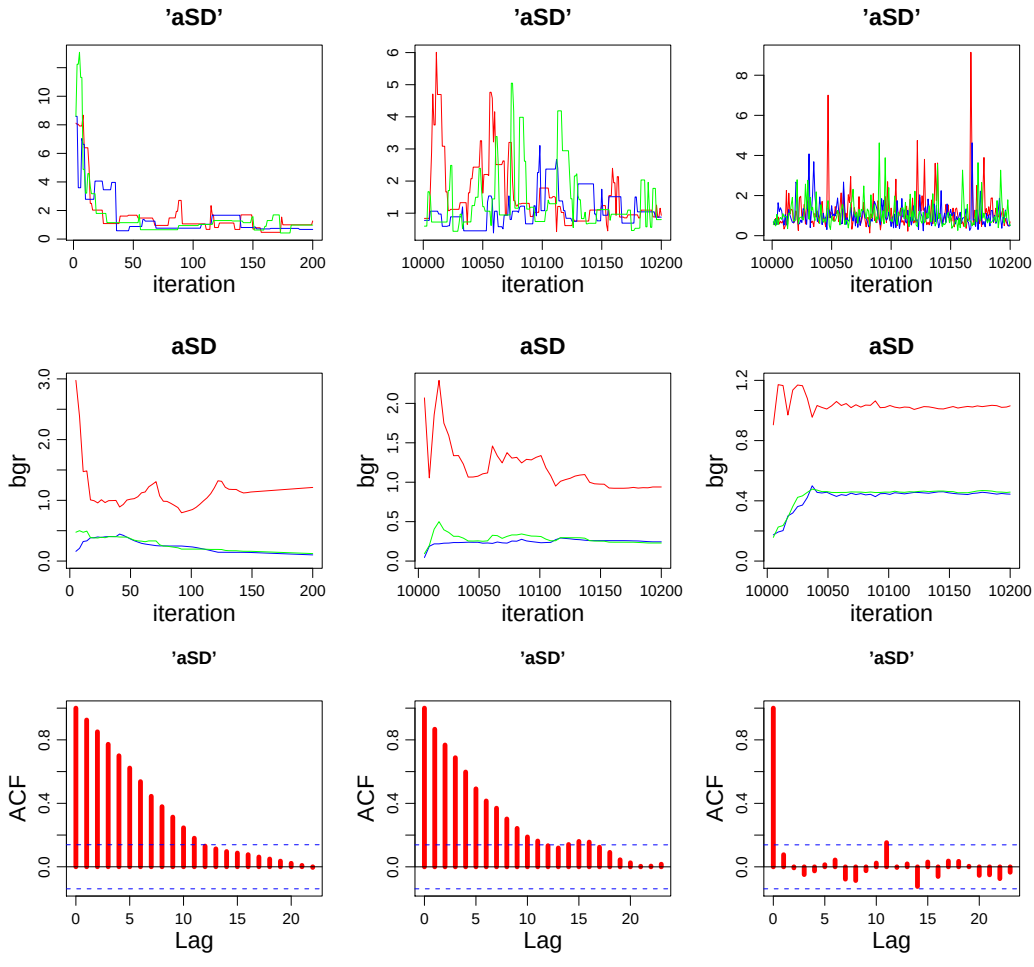

Figure 23.1: Effects of burn-in and thinning on MCMC chains. Left column: Zero burn-in steps and no thinning. Middle column: 10,000 burn-in steps and no thinning. Right column: 10,000 burn-in steps and thinning to 1 every 500 steps.

chains are actually exploring the bulk of the distribution, and are not stuck in some unrepresentative zone, we run several chains at once and examine whether they are well mixed, without any outlying chain.

One measure of mixing considers the variability between chains relative to the variability within chains. If the chains are well mixed, then this ratio should be about 1, i.e., there should be no more variability between chains than within chains. But if the chains are not well mixed, and one or more chains is lingering far from the rest, then the variance between chains will be larger than the variance within chains. This ratio is merely an F ratio for the chain values over a specified window of steps. In BUGS it is called the bgr statistic, after Brooks, Gelman and Rubin (see Brooks & Gelman, 1998).

To illustrate these ideas, consider the parameter from oneway ANOVA, near the top of the hierarchical diagram of Figure 18.1, p. 403. The standard deviation expresses the estimated variation across groups. In the BRugs program of Section 18.4.1 (ANOVAonewayBRugs.R), this parameter is denoted aSD. The top-left panel of Figure 23.1 shows the initial steps of three chains when they are intentionally started far from truly representa-

tive values. You can see that it takes about 50 steps (labeled as "iterations" in the graph) for the chains to descend to representative values. The top panel of Figure 23.1 shows the chains after 10,000 burn-in steps (which is overkill for this particular application). The chains appear to be converged, without systematically increasing or decreasing.

The bottom row of panels in Figure 23.1 shows the autocorrelation function, ACF. The left and middle columns show results when there is no thinning of the chains, and you can see that the ACF remains high for lags up to 15 or 20 steps. The correlation is visible in the chains themselves as sustained plateaus during which the values from step to step barely change. The right column shows results when the chains were thinned, keeping only 1 step in every 500. You can see that after even one of the thinned steps, the ACF(Lag=1) is nearly zero.

The middle row of Figure 23.1 shows the bgr statistic, which measures how well mixed the chains are, over a limited length of the chains. As mentioned above, the bgr should be around 1.0. The plots show the bgr as a curve (plotted as red by BUGS) hovering near a value of 1.0. The two other curves, lower in graphs (and plotted as green and blue by BUGS), show the between-chain and within-chain variances. For this particular application, the chains converge quickly and well.

The graphs in Figure 23.1 were produced by the following R program, which can be called after running any BUGS model. The argument `nodename` is a string that is name of any variable monitored by BUGS. (plotChains.R)

```
plotChains = function( nodename , saveplots=F , filenameroot="DeleteMe" ) {
    summarytable = samplesStats(nodename)
    show( summarytable )
    nCompon = NROW(summarytable)
    nPlotPerRow = 5
    nPlotRow = ceiling(nCompon/nPlotPerRow)
    nPlotCol = ceiling(nCompon/nPlotRow)
    windows(3.75*nPlotCol,3.5*nPlotRow)
    par( mar=c(4,4,3,1) , mgp=c(2,0.7,0) )
    samplesHistory( nodename , ask=F , mfrow=c(nPlotRow,nPlotCol) ,
                    cex.lab=1.5 , cex.main=1.5 )
    if ( saveplots ) {
      dev.copy2eps( file=paste( filenameroot , toupper(nodename) ,
                                "history.eps" , sep="" )) }
    windows(3.75*nPlotCol,3.5*nPlotRow)
    par( mar=c(4,4,3,1) , mgp=c(2,0.7,0) )
    samplesAutoC( nodename , chain=1 , ask=F , mfrow=c(nPlotRow,nPlotCol) ,
                  cex.lab=1.5 , cex.main=1.5 )
    if ( saveplots ) {
      dev.copy2eps( file=paste( filenameroot , toupper(nodename) ,
                                "autocorr.eps" , sep="" )) }
    windows(3.75*nPlotCol,3.5*nPlotRow)
    par( mar=c(4,4,3,1) , mgp=c(2,0.7,0) )
    samplesBgr( nodename , ask=F , mfrow=c(nPlotRow,nPlotCol) ,
                cex.lab=1.5 , cex.main=1.5 )
    if ( saveplots ) {
      dev.copy2eps( file=paste( filenameroot , toupper(nodename) ,
                                "bgr.eps" , sep="" )) }
    return( summarytable )
}
```

23.3 Functions for approximating highest density intervals

HDI's have been used routinely throughout the book to describe distributions. This section provides details regarding how the HDI's are computed. The algorithm for computing an HDI on a grid approximation applied to any dimensionality and any shape distribution. The algorithms for computing an HDI of an MCMC sample or for a function apply only to single parameters with unimodal distributions.

23.3.1 R code for computing HDI of a grid approximation

This function was first used in the R code of Section 6.7.1 (BernGrid.R), and again in Section 8.8.1 (BernTwoGrid.R).

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

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

(HDIofGrid.R)

```
HDIofGrid = function( probMassVec , credMass=0.95 ) {
    # Arguments:
    #   probMassVec is a vector of probability masses at each grid point.
    #   credMass is the desired mass of the HDI region.
    # Return value:
    #   A list with components:
    #   indices is a vector of indices that are in the HDI
    #   mass is the total mass of the included indices
    #   height is the smallest component probability mass in the HDI
    # Example of use: For determining HDI of a beta(30,12) distribution
    #   approximated on a grid:
    #   > probDensityVec = dbeta( seq(0,1,length=201) , 30 , 12 )
    #   > probMassVec = probDensityVec / sum( probDensityVec )
    #   > HDIinfo = HDIofGrid( probMassVec )
    #   > show( HDIinfo )
    sortedProbMass = sort( probMassVec , decreasing=T )
    HDIheightIdx = min( which( cumsum( sortedProbMass ) >= credMass ) )
    HDIheight = sortedProbMass[ HDIheightIdx ]
    HDImass = sum( probMassVec[ probMassVec >= HDIheight ] )
    return( list( indices = which( probMassVec >= HDIheight ) ,
                  mass = HDImass , height = HDIheight ) )
}
```

23.3.2 R code for computing HDI of a MCMC sample

The algorithms for computing the HDI for an MCMC sample or for a function rely on a crucial property: For a unimodal probability distribution on a single variable, the HDI of

mass M is the narrowest possible interval of that mass. Figure 23.2 illustrates why this is true. Consider the 90% HDI as shown. We construct another interval of 90% mass by moving the limits of the HDI to right, such that each limit is moved to a point that covers 4%, as marked in grey in Figure 23.2. The new interval does indeed cover 90%, because the 4% lost on the left is replaced by the 4% gained on the right.

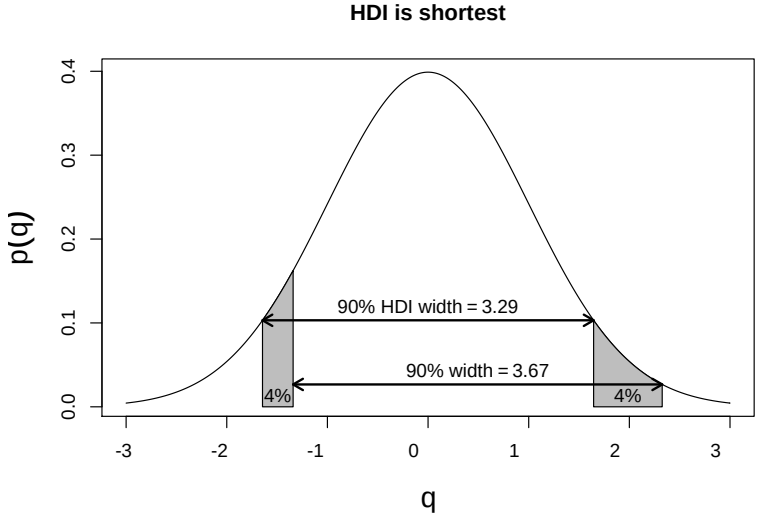

Figure 23.2: For a unimodal distribution, the HDI is the narrowest interval of that mass. This figure shows the 90% HDI and another interval that has 90% mass.

Consider the grey regions in Figure 23.2. Their left edges are the same height, because the left edges are de ned by the HDI. Their areas are the same, because, by de nition, the areas are both 4%. Notice, however, that the left grey area is taller than the right grey area, because the left edge falls at a point where the distribution is increasing in density, but the right edge falls at a point where the distribution is decreasing in density. Therefore, because the areas are the same but the left is taller than the right, the width of the left grey zone must be less than the width of the right grey zone. Consequently, the distance between right edges of the two grey zones must be greater than the HDI width. These widths are marked in Figure 23.2.

This argument applies for any size of grey zone, and for any size HDI. The argument relies on unimodality, however. Given the argument and diagram in Figure 23.2, it is not too hard to believe the converse: For a unimodal distribution on one variable, for any mass M, the interval containing mass M that has the narrowest width is the HDI for that mass. The algorithms described below are based on this property of HDIs. The algorithms nd the HDI by searching among candidate intervals of mass M. The shortest one found is declared to be the approximate HDI. It is an approximation, of course. (See Chen and Shao (1999) for more details, and Chen, He, Shao, and Xu (2003) for dealing with the unusual situation of multimodal distributions.)

Below is the code for nding the HDI of an MCMC sample. It is brief and hopefully self-explanatory after the discussion above.

(HDIofMCMC.R)

```
1 HDIofMCMC = function( sampleVec , credMass=0.95 ) {
2     # Computes highest density interval from a sample of represe    ntative values,
```

```
    #    estimated as shortest credible interval.
    # Arguments:
    #    sampleVec
    #       is a vector of representative values from a probability dis        tribution.
    #    credMass
    #       is a scalar between 0 and 1, indicating the mass within the cr      edible
    #       interval that is to be estimated.
    # Value:
    #    HDIlim is a vector containing the limits of the HDI
    sortedPts = sort( sampleVec )
    ciIdxInc = floor( credMass * length( sortedPts ) )
    nCIs = length( sortedPts ) - ciIdxInc
    ciWidth = rep( 0 , nCIs )
    for ( i in 1:nCIs ) {
        ciWidth[ i ] = sortedPts[ i + ciIdxInc ] - sortedPts[ i ]
    }
    HDImin = sortedPts[ which.min( ciWidth ) ]
    HDImax = sortedPts[ which.min( ciWidth ) + ciIdxInc ]
    HDIlim = c( HDImin , HDImax )
    return( HDIlim )
}
```

23.3.3 R code for computing HDI of a function

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

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

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

(HDIofICDF.R)

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

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

23.4 Reparameterization of probability distributions

There are situations in which it is natural to express a distribution on one scale, but parameterize it mathematically on a different scale. For example, we may think intuitively of the standard deviation of a normal distribution, but have to parameterize it in terms of the precision (i.e., reciprocal of the variance). As another example, we may think intuitively of an underlying bias on an infinite scale, but have to parameterize it on a zero-to-one interval. The question is, if we express a probability distribution on one scale, what is the equivalent distribution on a transformed scale?

The answer is not difficult to figure out, especially for single parameters. Let the "destination" parameter be denoted γ, and suppose that $\gamma = f(\theta)$ for the "source" parameter θ, with a monotonic and differentiable function f. Let the probability distribution on θ be denoted $p(\theta)$. Then the corresponding probability distribution on γ is $p(\gamma) = p(f^{-1}(\gamma)) \left| f'(f^{-1}(\gamma)) \right|$, where $f'(\theta)$ is the derivative of f with respect to θ.

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

23.4.1 Examples

We can apply the general formula to the case in which $\gamma = f(\theta) = \mathrm{sig}(\theta) = 1/[1+\exp(-\theta)]$, and the distribution on θ is $p(\theta) = (f(\theta))^a (1-f(\theta))^b / B(a,b) = \gamma^a (1-\gamma)^b / B(a,b)$. Notice that the derivative of f is $f'(\theta) = \exp(-\theta)/[1+\exp(-\theta)]^2 = \mathrm{sig}(\theta)(1-\mathrm{sig}(\theta)) = \gamma(1-\gamma)$. Therefore, the equivalent probability density at $\gamma = f(\theta)$ is $p(\gamma) = p(\theta)/f'(\theta) = \gamma^a (1-\gamma)^b / [\gamma(1-\gamma)B(\theta,a,b)] = \gamma^{(a-1)} (1-\gamma)^{(b-1)} / B(a,b) = \mathrm{beta}(\gamma,a,b)$. The upper row of Figure 23.3 shows this situation when $a = b = 1$. An intuitive way to think of this situation is that the probability on θ is dense near $\theta = 0$, but that is exactly where the sigmoidal transformation stretches the distribution. On the other hand, the probability on θ is sparse at large positive or large negative values, but that is exactly where the sigmoidal

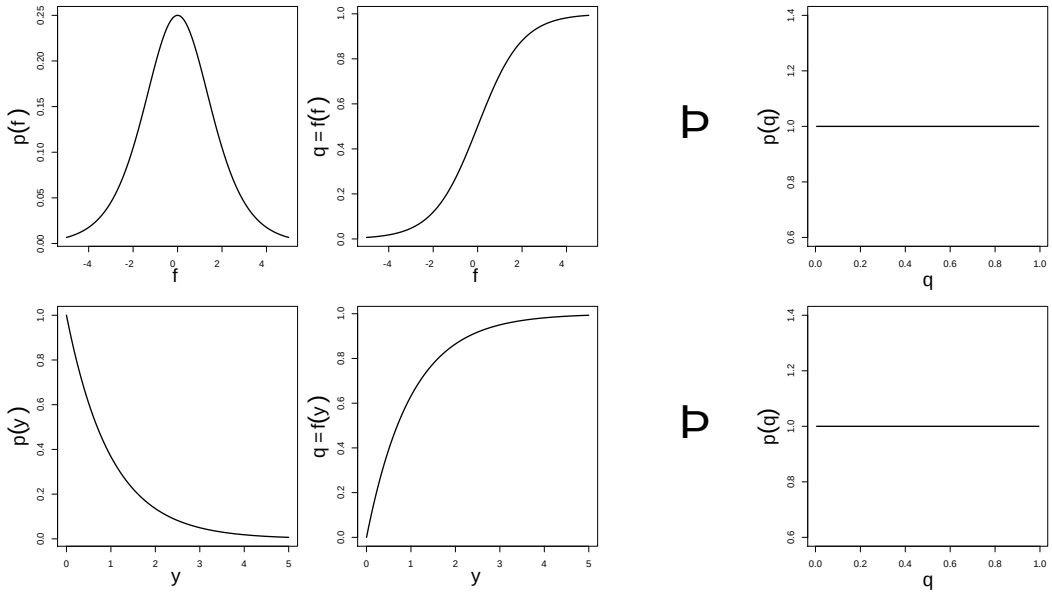

Figure 23.3: Top row shows a reparameterization that maps $[-1, +1]$ to $[0, 1]$. Bottom row shows a reparameterization that maps $[0, +\infty]$ to $[0, 1]$.

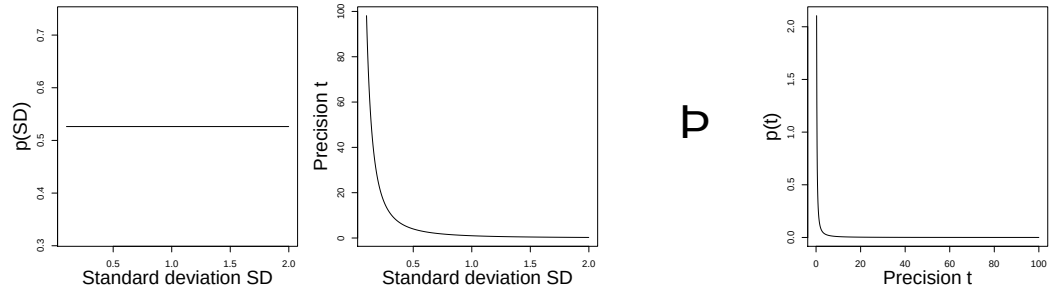

Figure 23.4: A uniform distribution on standard deviation, mapped to the corresponding distribution on precision.

transformation compresses the distribution.

As another example, consider a case in which $\theta = f(\phi) = 1 - \exp(-\phi)$, with the probability density $p(\phi) = \exp(-\phi)$. Notice that the derivative of the transformation is $f'(\phi) = \exp(-\phi)$, and therefore the equivalent density at $\theta = f(\phi)$ is $p(f(\phi)) = p(\phi)/f'(\phi) = 1$. In other words, the equivalent density on θ is the uniform density, as shown in the lower panel of Figure 23.3.

As a final example, Figure 23.4 shows a uniform distribution on standard deviation transformed to the corresponding distribution on precision. By definition, precision is the reciprocal of squared standard deviation.

23.4.2 Reparameterization of two parameters

When there is more than one parameter being transformed, the calculus becomes a little more involved. Suppose we have a probability density on a parameter space $p(\theta_1, \theta_2)$. Let $\theta_1 = f_1(\phi_1, \phi_2)$ and $\theta_2 = f_2(\phi_1, \phi_2)$. Our goal is to find the probability density $p(\phi_1, \phi_2)$

that corresponds to $p(\theta_1, \theta_2)$. We do this by considering infinitesimal corresponding regions. Consider a point θ_1, θ_2. The probability mass of a small region near that point is the density at that point times the area of the small region: $p(\theta_1, \theta_2)\, d\theta_1\, d\theta_2$. The corresponding region in the transformed parameters should have the same mass. That mass is the density at the transformed point times the area of the region mapped to from the originating region. In vector calculus textbooks, you can find discussions demonstrating that the area of the mapped-to region is $|\det(J)|\, d\theta_1\, d\theta_2$ where J is the Jacobian matrix: $J_{rc} = d f_r(\theta_1, \theta_2)/d\theta_c$ and $\det(J)$ is the determinant of the Jacobian matrix. Setting the two masses equal and re-arranging yields $p(\theta_1, \theta_2) = p(\theta_1, \theta_2)/|\det(J)|$.

As we have had no occasions to apply this transformation, no examples will be provided here. But the method has been mentioned for those intrepid souls who may wish to venture into the wilderness of multivariate probability distributions with nothing more than pen and paper.

References

Adcock, C. J. (1997). Sample size determination: a review. *The Statistician*, *46*, 261–283.

Agresti, A., & Hitchcock, D. B. (2005). Bayesian inference for categorical data analysis. *Statistical Methods & Applications*, *14*(3), 297–330.

Albert, J. H., & Rossman, A. J. (2001). *Workshop statistics: Discovery with data, a Bayesian approach*. Emeryville, CA: Key College Publishing.

Berger, J. O. (1985). *Statistical decision theory and Bayesian analysis, 2nd ed.* New York: Springer.

Berger, J. O., & Berry, D. A. (1988). Statistical analysis and the illusion of objectivity. *American Scientist*, *76*(2), 159–165.

Berger, R. L., Boos, D. D., & Guess, F. M. (1988). Tests and confidence sets for comparing two mean residual life functions. *Biometrics*, *44*(1), 103–115.

Berry, D. A. (1996). *Statistics: A Bayesian perspective*. Belmont, CA: Duxbury Press/Wadsworth.

Berry, D. A., & Hochberg, Y. (1999). Bayesian perspectives on multiple comparisons. *Journal of Statistical Planning and Inference*, *82*(1-2), 215–227.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.

Bliss, C. I. (1934). The method of probits. *Science*, *79*(2037), 38–39.

Bolstad, W. M. (2007). *Introduction to Bayesian statistics* (2nd ed.). Hoboken, NJ: Wiley.

Brambor, T., Clark, W. R., & Golder, M. (2006). Understanding interaction models: Improving empirical analyses. *Political Analysis*, *14*, 63–82.

Braumoeller, B. F. (2004). Hypothesis testing and multiplicative interaction terms. *International Organization*, *58*(04), 807–820.

Brehmer, B. (1974). Hypotheses about relations between scaled variables in the learning of probabilistic inference tasks. *Organizational Behavior and Human Performance*, *11*, 1–27.

Brooks, S. P., & Gelman, A. (1998). Alternative methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, *7*, 434–455.

Carlin, B. P., & Chib, S. (1995). Bayesian model choice via Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society, Series B*, *57*(3), 473–484.

Carlin, B. P., & Louis, T. A. (2000). *Bayes and empirical Bayes methods for data analysis* (2nd ed.). Boca Raton, FL: Chapman & Hall/CRC.

Carlin, B. P., & Louis, T. A. (2009). *Bayesian methods for data analysis* (3rd ed.). Boca Raton, FL: CRC Press.

Casella, G., & Moreno, E. (2006). Objective Bayesian variable selection. *Journal of the American Statistical Association, 101*(473), 157–167.

Chaloner, K., & Verdinelli, I. (1995). Bayesian experimental design: A review. *Statistical Science, 10*(3), 273–304.

Chen, M.-H., He, X., Shao, Q.-M., & Xu, H. (2003). A Monte Carlo approach in computing HPD regions. *Development of Modern Statistics and Related Topics: In Celebration of Professor Yaoting Zhang's 70th Birthday*, 38–52.

Chen, M. H., & Shao, Q. M. (1999). Monte Carlo estimation of Bayesian credible and HPD intervals. *Journal of Computational and Graphical Statistics, 8*, 69–92.

Clyde, M., & George, E. I. (2004). Model uncertainty. *Statistical Science, 19*, 81–94.

Congdon, P. (2005). *Bayesian models for categorical data*. West Sussex, England: Wiley.

Damgaard, L. H. (2007). Technical note: How to use WinBUGS to draw inferences in animal models. *Journal of Animal Science, 85*, 1363–1368.

Dawes, J. (2008). Do data characteristics change according to the number of scale points used? an experiment using 5-point, 7-point and 10-point scales. *International Journal of Market Research, 50*(1), 61–77.

de Saint-Exupery, A. (1943). *The little prince* San Diego: Harcourt.

De Santis, F. (2004). Statistical evidence and sample size determination for Bayesian hypothesis testing. *Journal of Statistical Planning and Inference, 124*, 121–144.

De Santis, F. (2007). Using historical data for Bayesian sample size determination. *Journal of the Royal Statistical Society: Series A, 170*, 95–113.

DeGroot, M. H. (2004). *Optimal statistical decisions*. New York: Wiley Interscience.

Edwards, W., Lindman, H., & Savage, L. J. (1963). Bayesian statistical inference for psychological research. *Psychological Review, 70*, 193–242.

Feldman, H. A. (1988). Families of lines: random effects in linear regression analysis. *Journal of Applied Physiology, 64*(4), 1721–1732.

Freedman, L. S., Lowe, D., & Macaskill, P. (1984). Stopping rules for clinical trials incorporating clinical opinion. *Biometrics, 40*, 575–586.

Gallistel, C. R. (2009). The importance of proving the null. *Psychological Review, 116*(2), 439–453.

Gelfand, A. E., & Dey, D. K. (1994). Bayesian model choice: asymptotics and exact calculations. *Journal of the Royal Statistical Society, Series B, 56*, 501–514.

Gelman, A. (2005). Analysis of variance — why it is more important than ever. The Annals of Statistics, 33(1), 1–53.

Gelman, A. (2006). Prior distributions for variance parameters in hierarchical models. Bayesian Analysis, 1(3), 515–533.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). Bayesian data analysis (2nd ed.). Boca Raton, Florida: CRC Press.

Gelman, A., & Hill, J. (2007). Data analysis using regression and multilevel/hierarchical models New York: Cambridge University Press.

Gelman, A., Hill, J., & Yajima, M. (2009). Why we (usually) don't have to worry about multiple comparisons. Available from http://www.stat.columbia.edu/ gelman/research/unpublished/multiple2.pdf

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 6, 721–741.

George, D. N., & Pearce, J. M. (1999). Acquired distinctiveness is controlled by stimulus relevance not correlation with reward. Journal of Experimental Psychology: Animal Behavior Processes, 25(3), 363–373.

George, E. I. (2000). The variable selection problem. Journal of the American Statistical Association, 95(452).

Gigerenzer, G., & Horage, U. (1995). How to improve Bayesian reasoning without instruction: Frequency formats. Psychological Review, 102, 684-704.

Gigerenzer, G., Krauss, S., & Vitouch, O. (2004). The null ritual: What you always wanted to know about signi cance testing but were afraid to ask. In D. Kaplan (Ed.), The Sage handbook of quantitative methodology for the social sciences (pp. 391–408). Thousand Oaks, CA: Sage.

Gilks, W. R., Thomas, A., & Spiegelhalter, D. J. (1994). A language and program for complex Bayesian modelling. The Statistician, 43(1), 169–177.

Gill, J. (2002). Bayesian methods for the social and behavioral sciences. Boca Raton, Florida: CRC Press.

Gilovich, T., Vallone, R., & Tversky, A. (1985). The hot hand in basketball: On the misperception of random sequences. Cognitive Psychology, 17(3), 295–314.

Gopalan, R., & Berry, D. (1998). Bayesian multiple comparisons using Dirichlet process priors. Journal of the American Statistical Association, 1130–1139.

Gosset, W. S. (1908). The probable error of a mean. Biometrika, 6, 1–25.

Greenland, S. (2008). Invited commentary: variable selection versus shrinkage in the control of multiple confounders. American Journal of Epidemiology, 167(5), 523–529.

Guber, D. L. (1999). Getting what you pay for: The debate over equity in public school expenditures. Journal of Statistics Education, 7(2). Available from http://www.amstat.org/publications/JSE/secure/v7n2/datasets.guber.cfm

Hahn, U., Chater, N., & Richardson, L. B. (2003). Similarity as transformation. Cognition, 87(1), 1–32.

Han, C., & Carlin, B. P. (2001). Markov chain Monte Carlo methods for computing Bayes factors: A comparative review. Journal of the American Statistical Association, 96(455), 1122–1132.

Hand, D. J., Daly, F., Lunn, A. D., McConway, K. J., & Ostrowski, E. (1994). A handbook of small data sets. London: Chaprman & Hall.

Hobbs, B. P., & Carlin, B. P. (2008, January). Practical Bayesian design and analysis for drug and device clinical trials. Journal of Biopharmaceutical Statistics, 18(1), 54–80.

Hoffman, P. J., Earle, T. C., & Slovic, P. (1981). Multidimensional functional learning (MFL) and some new conceptions of feedback. Organizational Behavior and Human Performance, 27(1), 75–102.

Holcomb, J., & Spalsbury, A. (2005). Teaching students to use summary statistics and graphics to clean and analyze data. Journal of Statistics Education, 13(3). Available from http://www.amstat.org/publications/jse/v13n3/datasets.holcomb.html

Hyndman, R. J. (1996). Computing and graphing highest density regions. The American statistician, 50(2), 120–126.

Joseph, L., Wolfson, D. B., & du Berger, R. (1995a). Sample size calculations for binomial proportions via highest posterior density intervals. The Statistician, 44, 143–154.

Joseph, L., Wolfson, D. B., & du Berger, R. (1995b). Some comments on Bayesian sample size determination. The Statistician, 44, 167–171.

Kalish, M. L., Griffiths, T. L., & Lewandowsky, S. (2007). Iterated learning: Intergenerational knowledge transmission reveals inductive biases. Psychonomic Bulletin & Review, 14(2), 288.

Kass, R. E., & Raftery, A. E. (1995). Bayes factors. Journal of the American Statistical Association, 90, 773–795.

Keith, T. (2005). Multiple regression and beyond. Columbus, OH: Allyn & Bacon.

Kolmogorov, A. N. (1956). Foundations of the theory of probability. New York: Chelsea.

Krauss, S., Martignon, L., & Horrage, U. (1999). Simplifying Bayesian inference: The general case. In L. Magnani, N. J. Nersessian, & P. Thagards (Eds.), Model-based reasoning in scientific discovery (pp. 165–180). New York: Springer.

Kruschke, J. K. (1993). Human category learning: Implications for backpropagation models. Connection Science, 5, 3–36.

Kruschke, J. K. (1996). Dimensional relevance shifts in category learning. *Connection Science*, *8*, 201–223.

Kruschke, J. K. (2008). Bayesian approaches to associative learning: From passive to active learning. *Learning & Behavior*, *36*(3), 210-226.

Kruschke, J. K. (2009). Highlighting: A canonical experiment. In B. Ross (Ed.), *The psychology of learning and motivation* (Vol. 51, pp. 153–185). Elsevier Academic Press.

Kruschke, J. K. (2010). Bayesian data analysis. *Wiley Interdisciplinary Reviews: Cognitive Science*, ** (**), **-**.

Learner, E. E. (1978). *Specification searches*. New York: Wiley.

Lee, M. D., & Webb, M. R. (2005). Modeling individual differences in cognition. *Psychonomic Bulletin & Review*, *12*(4), 605–621.

Liang, F., Paulo, R., Molina, G., Clyde, M. A., & Berger, J. O. (2008). Mixtures of g priors for Bayesian variable selection. *Journal of the American Statistical Association*, *103*, 410–423.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, *140*, 1–55.

Lindley, D. V. (1997). The choice of sample size. *The Statistician*, *46*, 129–138.

Lindley, D. V., & Phillips, L. D. (1976). Inference for a Bernoulli process (a Bayesian view). *The American Statistician*, *30*(3), 112–119.

Lindquist, M. A., & Gelman, A. (2009). Correlations and multiple comparisons in functional imaging – a statistical perspective. *Perspectives in Psychological Science*, *4*(3), 310–313.

Liu, C. C., & Aitkin, M. (2008). Bayes factors: Prior sensitivity and model generalizability. *Journal of Mathematical Psychology*, *52*, 362–375.

Lynch, S. M. (2007). *Introduction to applied Bayesian statistics and estimation for social scientists*. New York: Springer.

MacKay, D. J. C. (2003). *Information theory, inference & learning algorithms*. Cambridge, UK: Cambridge University Press.

Marin, J.-M., & Robert, C. P. (2007). *Bayesian core: A practical approach to computational Bayesian statistics*. New York: Springer.

Maxwell, S. E., & Delaney, H. D. (2004). *Designing experiments and analyzing data: a model comparison perspective* (2nd ed.). Mahwah, NJ: Erlbaum.

McCullagh, P., & Nelder, J. (1989). *Generalized linear models, 2nd ed*. Boca Raton, FL: Chapman and Hall/CRC.

McDonald, J. H. (2009). *Handbook of biological statistics (2nd ed.)*. Baltimore, Maryland: Sparky House Publishing.

McDonald, J. H., Seed, R., & Koehn, R. K. (1991). Allozymes and morphometric characters of three species of Mytilus in the Northern and Southern hemispheres. Marine Biology, 111(3), 323–333.

McIntyre, L. (1994). Using cigarette data for an introduction to multiple regression. Journal of Statistics Education, 2(1). Available from http://www.amstat.org/publications/jse/v2n1/datasets.mcintyre.html

Meng, C. Y. K., & Dempster, A. P. (1987). A Bayesian approach to the multiplicity problem for significance testing with binomial data. Biometrics, 43(2), 301–311.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equations of state calculations by fast computing machines. Journal of Chemical Physics, 21, 1087–1091.

Meyer, R., & Yu, J. (2000). BUGS for a Bayesian analysis of stochastic volatility models. Econometrics Journal, 3(2), 198–215.

Miller, J. (2009). What is the probability of replicating a statistically significant effect? Psychonomic Bulletin & Review, 16(4), 617–640.

Moore, T. L. (2006). Paradoxes in film ratings. Journal of Statistics Education, 14(1). Available from www.amstat.org/publications/jse/v14n1/datasets.moore.html

Mueller, P., Parmigiani, G., & Rice, K. (2007). FDR and Bayesian multiple comparisons rules. In J. M. Bernardo et al. (Eds.), Bayesian statistics 8. Oxford, UK: Oxford University Press.

Navarro, D. J., Griffiths, T. L., Steyvers, M., & Lee, M. D. (2006). Modeling individual differences using Dirichlet processes. Journal of Mathematical Psychology, 50, 101–122.

Nelder, J. A., & Wedderburn, R. W. M. (1972). Generalized linear models. Journal of the Royal Statistical Society. Series A (General), 135(3), 370–384.

Nietzsche, F. (1967). The will to power. New York: Random House. (Translated by W. Kaufmann and R. J. Hollingdale)

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

Oswald, C. J. P., Yee, B. K., Rawlins, J. N. P., Bannerman, D. M., Good, M., & Honey, R. C. (2001). Involvement of the entorhinal cortex in a process of attentional modulation: Evidence from a novel variant of an IDED procedure. Behavioral neuroscience, 115(4), 841–849.

Pham-Gia, T., & Turkkan, N. (1992). Sample size determination in Bayesian analysis. The Statistician, 41, 389–392.

Poldrack, R. A. (2006). Can cognitive processes be inferred from neuroimaging data? Trends in Cognitive Sciences, 10(2), 59–63.

Proschan, F. (1963). Theoretical explanation of observed decreasing failure rate. Technometrics, 375–383.

Qian, S. S., & Shen, Z. (2007). Ecological applications of the three-level analysis of variance. *Ecology, 88*(10), 2489–2495.

Robert, C. P., & Casella, G. (2004). *Monte Carlo statistical methods* (2nd ed.). New York: Springer.

Rosa, L., Rosa, E., Sarner, L., & Barrett, S. (1998). A close look at therapeutic touch. *Journal of the American Medical Association, 279*(13), 1005–1010.

Rouder, J. N., & Lu, J. (2005). An introduction to Bayesian hierarchical models with an application in the theory of signal detection. *Psychonomic Bulletin & Review, 12*(4), 573–604.

Rouder, J. N., Lu, J., Speckman, P., Sun, D., & Jiang, Y. (2005). A hierarchical model for estimating response time distributions. *Psychonomic Bulletin & Review, 12*(2), 195–223.

Rouder, J. N., Speckman, P. L., Sun, D., Morey, R. D., & Iverson, G. (2009). Bayesian t-tests for accepting and rejecting the null hypothesis. *Psychonomic Bulletin & Review, 16*, 225–237.

Roy, A., Ghosal, S., & Rosenberger, W. F. (2009). Convergence properties of sequential Bayesian D-optimal designs. *Journal of Statistical Planning and Inference, 139*, 425–440.

Sadiku, M. N. O., & To ghi, M. R. (1999). A tutorial on simulation of queueing models. *International Journal of Electrical Engineering Education, 36*, 102–120.

Scott, J. G., & Berger, J. O. (2006). An exploration of aspects of Bayesian multiple testing. *Journal of statistical planning and inference, 136*(7), 2144–2162.

Snee, R. D. (1974). Graphical display of two-way contingency tables. *The American Statistician, 28*(1), 9–12.

Solari, F., Liseo, B., & Sun, D. (2008). Some remarks on Bayesian inference for one-way ANOVA models. *Annals of the Institute of Statistical Mathematics, 60*, 483–498.

Spiegelhalter, D. J., Freedman, L. S., & Parmar, M. K. B. (1994). Bayesian approaches to randomized trials. *Journal of the Royal Statistical Society. Series A, 157*, 357–416.

Stevens, S. S. (1946). On the theory of scales of measurement. *Science, 103*(2684), 677–680.

Thomas, A. (2004). *BRugs user manual (the R interface to BUGS)*. Available from http://mathstat.helsinki.fi/openbugs/data/Docu/BRugsManual.html

Thomas, A., O'Hara, B., Ligges, U., & Sturtz, S. (2006, March). Making BUGS open. *R News, 6*(1), 12–17.

Tsionas, E. G. (2002). Bayesian inference in the noncentral Student-t model. *Journal of Computational and Graphical Statistics, 11*(1), 208–221.

Wagenmakers, E. J. (2007). A practical solution to the pervasive problems of p values. *Psychonomic Bulletin & Review, 14*(5), 779–804.

Walker, L. J., Gustafson, P., & Frimer, J. A. (2007). The application of Bayesian analysis to issues in developmental research. International Journal of Behavioral Development, 31(4), 366.

Wang, F., & Gelfand, A. E. (2002). A simulation-based approach to Bayesian sample size determination for performance under a given model and for separating models. Statistical Science, 17, 193–208.

Weiss, R. (1997). Bayesian sample size calculations for hypothesis testing. The Statistician, 46, 185–191.

Werner, M., Stabenau, J. R., & Pollin, W. (1970). Thematic apperception test method for the differentiation of families of schizophrenics, delinquents, and "normals". Journal of Abnormal Psychology, 75(2), 139–145.

Western, B., & Jackman, S. (1994). Bayesian inference for comparative research. The American Political Science Review, 88(2), 412–423.

Winer, B. J., Brown, D. R., & Michels, K. M. (1991). Statistical principles in experimental design, 3rd ed. New York: McGraw-Hill.

The Index on the following pages is incomplete. It includes only a few items to test the indexing facility. The index will be expanded at a later date.

Index