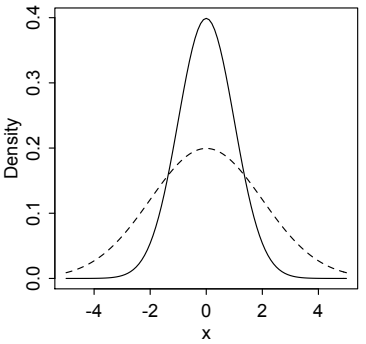
13. Chapter 13 Solutions

13E1. Option (a) will produce more shrinkage, because the prior is more concentrated. If this isn't obvious from the smaller standard deviation, you can always plot the two priors and compare:

R code 13.1

```
curve( dnorm(x,0,1) , from=-5 , to=5 , ylab="Density" )
curve( dnorm(x,0,2) , add=TRUE , lty=2 )
```



Since option (a), shown by the solid density, piles up more mass around zero, it will pull extreme values closer to zero.

13E2. All that is really required to convert the model to a multilevel model is to take the prior for the vector of intercepts, α_{GROUP}, and make it adaptive. This means we define parameters for its mean and standard deviation. Then we assign these two new parameters their own priors, *hyperpriors*. This is what it looks like:

$$\begin{aligned} y_i &\sim \text{Binomial}(1, p_i) \\ \text{logit}(p_i) &= \alpha_{\text{GROUP}[i]} + \beta x_i \\ \alpha_{\text{GROUP}} &\sim \text{Normal}(\bar{\alpha}, \sigma_\alpha) \\ \beta &\sim \text{Normal}(0, 1) \\ \bar{\alpha} &\sim \text{Normal}(0, 1.5) \\ \sigma_\alpha &\sim \text{Exponential}(1) \end{aligned}$$

The exact hyperpriors you assign don't matter here. Since this problem has no data context, it isn't really possible to say what sensible priors would be. Note also that an exponential prior on σ_α is just as sensible, absent context, as the half-Cauchy prior.

13E3. This is very similar to the previous problem. The only trick here is to notice that there is already a standard deviation parameter, σ. But that standard deviation is for the *residuals*, at the top

level. We'll need yet another standard deviation for the varying intercepts:

$$\begin{aligned} y_i &\sim \text{Normal}(\mu_i, \sigma) \\ \mu_i &= \alpha_{\text{GROUP}[i]} + \beta x_i \\ \alpha_{\text{GROUP}} &\sim \text{Normal}(\bar{\alpha}, \sigma_\alpha) \\ \beta &\sim \text{Normal}(0, 1) \\ \sigma &\sim \text{Exponential}(0, 1) \\ \bar{\alpha} &\sim \text{Normal}(0, 5) \\ \sigma_\alpha &\sim \text{Exponential}(0, 1) \end{aligned}$$

13E4. You can just copy the answer from problem **13E2** and swap out the binomial likelihood for a Poisson, taking care to change the link function from logit to log. Of course you'd need to rethink the priors, and those will depend upon what the outcome is. But here is the right structure:

$$\begin{aligned} y_i &\sim \text{Poisson}(\lambda_i) \\ \log(\lambda_i) &= \alpha_{\text{GROUP}[i]} + \beta x_i \\ \alpha_{\text{GROUP}} &\sim \text{Normal}(\bar{\alpha}, \sigma_\alpha) \\ \beta &\sim \text{Normal}(0, 1) \\ \bar{\alpha} &\sim \text{Normal}(0, 1) \\ \sigma_\alpha &\sim \text{Exponential}(0, 1) \end{aligned}$$

Under the hood, all multilevel models are alike. It doesn't matter which likelihood function rests at the top. Take care, however, to reconsider priors. The scale of the data and parameters is likely quite different for a Poisson model. Absent any particular context in this problem, you can't recommend better priors. But in real work, it's good to think about reasonable values and provide regularizing priors on the relevant scale.

13E5. The cross-classified model adds another varying intercept type. This is no harder than duplicating the original varying intercepts structure. But you have to take care now not to over-parameterize the model by having a hyperprior mean for both intercept types. You can do this by just assigning one of the adaptive priors a mean of zero. Suppose for example that the second cluster type is DAY:

$$\begin{aligned} y_i &\sim \text{Poisson}(\lambda_i) \\ \log(\lambda_i) &= \alpha_{\text{GROUP}[i]} + \alpha_{\text{DAY}[i]} + \beta x_i \\ \alpha_{\text{GROUP}} &\sim \text{Normal}(\bar{\alpha}, \sigma_{\text{GROUP}}) \\ \alpha_{\text{DAY}} &\sim \text{Normal}(0, \sigma_{\text{DAY}}) \\ \beta &\sim \text{Normal}(0, 1) \\ \bar{\alpha} &\sim \text{Normal}(0, 1) \\ \sigma_{\text{GROUP}} &\sim \text{Exponential}(0, 1) \\ \sigma_{\text{DAY}} &\sim \text{Exponential}(0, 1) \end{aligned}$$

Or you can just pull the mean intercept out of both priors and put it in the linear model:

$$
\begin{aligned}
y_i &\sim \text{Poisson}(\lambda_i) \\
\log(\lambda_i) &= \bar{\alpha} + \alpha_{\text{GROUP}[i]} + \alpha_{\text{DAY}[i]} + \beta x_i \\
\alpha_{\text{GROUP}} &\sim \text{Normal}(0, \sigma_{\text{GROUP}}) \\
\alpha_{\text{DAY}} &\sim \text{Normal}(0, \sigma_{\text{DAY}}) \\
\beta &\sim \text{Normal}(0, 1) \\
\bar{\alpha} &\sim \text{Normal}(0, 1) \\
\sigma_{\text{GROUP}} &\sim \text{Exponential}(0, 1) \\
\sigma_{\text{DAY}} &\sim \text{Exponential}(0, 1)
\end{aligned}
$$

These are exactly the same model. Although as you'll see later in Chapter 14, these different forms might be more or less efficient in sampling.

13M1. First, let's set up the data list:

R code 13.2

```
library(rethinking)
data(reedfrogs)
d <- reedfrogs

dat <- list(
    S = d$surv,
    n = d$density,
    tank = 1:nrow(d),
    pred = ifelse( d$pred=="no" , 0L , 1L ),
    size_ = ifelse( d$size=="small" , 1L , 2L )
)
```

Now to define a series of models. The first is just the varying intercepts model from the text:

R code 13.3

```
m1.1 <- ulam(
    alist(
        S ~ binomial( n , p ),
        logit(p) <- a[tank],
        a[tank] ~ normal( a_bar , sigma ),
        a_bar ~ normal( 0 , 1.5 ),
        sigma ~ exponential( 1 )
    ), data=dat , chains=4 , cores=4 , log_lik=TRUE )
```

The other models just incorporate the predictors, as ordinary regression terms.

R code 13.4

```
# pred
m1.2 <- ulam(
    alist(
        S ~ binomial( n , p ),
        logit(p) <- a[tank] + bp*pred,
        a[tank] ~ normal( a_bar , sigma ),
        bp ~ normal( -0.5 , 1 ),
        a_bar ~ normal( 0 , 1.5 ),
        sigma ~ exponential( 1 )
    ), data=dat , chains=4 , cores=4 , log_lik=TRUE )
```

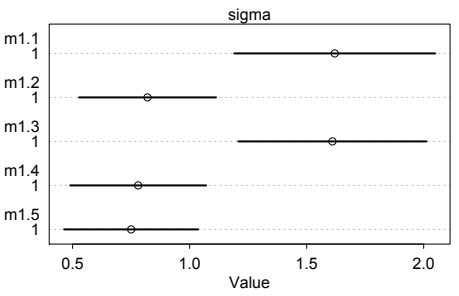
```
# size
m1.3 <- ulam(
    alist(
        S ~ binomial( n , p ),
        logit(p) <- a[tank] + s[size_],
        a[tank] ~ normal( a_bar , sigma ),
        s[size_] ~ normal( 0 , 0.5 ),
        a_bar ~ normal( 0 , 1.5 ),
        sigma ~ exponential( 1 )
    ), data=dat , chains=4 , cores=4 , log_lik=TRUE )

# pred + size
m1.4 <- ulam(
    alist(
        S ~ binomial( n , p ),
        logit(p) <- a[tank] + bp*pred + s[size_],
        a[tank] ~ normal( a_bar , sigma ),
        bp ~ normal( -0.5 , 1 ),
        s[size_] ~ normal( 0 , 0.5 ),
        a_bar ~ normal( 0 , 1.5 ),
        sigma ~ exponential( 1 )
    ), data=dat , chains=4 , cores=4 , log_lik=TRUE )

# pred + size + interaction
m1.5 <- ulam(
    alist(
        S ~ binomial( n , p ),
        logit(p) <- a_bar + z[tank]*sigma + bp[size_]*pred + s[size_],
        z[tank] ~ normal( 0 , 1 ),
        bp[size_] ~ normal( -0.5 , 1 ),
        s[size_] ~ normal( 0 , 0.5 ),
        a_bar ~ normal( 0 , 1.5 ),
        sigma ~ exponential( 1 )
    ), data=dat , chains=4 , cores=4 , log_lik=TRUE )
```

I coded the interaction model using a non-centered parameterization. The interaction itself is done by creating a `bp` parameter for each size value. In this way, the effect of pred depends upon size.

Let's look at all the `sigma` posterior distributions:

R code 13.5

```
plot( coeftab( m1.1 , m1.2 , m1.3 , m1.4 , m1.5 ), pars="sigma" )
```



The two models that omit predation, `m1.1` and `m1.3`, have larger values of sigma. This is because predation explains some of the variation among tanks. So when you add it to the model, the variation in the tank intercepts gets smaller. We'll examine this in more detail in a the last problem of this chapter.

The general point here is that the model with only intercepts measures the variation among tanks, but does nothing to explain it. As we add treatment variables, the variation should shrink, even though the total variation in the data of course stays the same.

13M2. The WAIC scores:

R code 13.6

```
compare( m1.1 , m1.2 , m1.3 , m1.4 , m1.5 )
```

```
      WAIC   SE dWAIC  dSE pWAIC weight
m1.2 198.8 8.97   0.0   NA  19.1   0.31
m1.1 199.2 7.10   0.4 5.65  20.5   0.25
m1.5 199.7 8.96   0.9 3.22  19.1   0.20
m1.4 200.3 8.83   1.4 2.13  19.2   0.15
m1.3 201.1 7.28   2.2 5.61  21.4   0.10
```

These models are really very similar in expected out-of-sample accuracy. The tank variation is huge. But take a look at the posterior distributions for predation and size. You'll see that predation does seem to matter, as you'd expect. Size matters a lot less. So while predation doesn't explain much of the total variation, there is plenty of evidence that it is a real effect. Remember: We don't select a model using WAIC (or PSIS). A predictor can make little difference in total accuracy but still be a real causal effect.

If you inspect the posterior distributions, you'll see that the coefficients for predation are further from zero than are the coefficients for size. This is consistent with the model rankings.

The fact that the tank-only model does so well does not mean that predation and size do not matter. The posterior distributions clearly suggest that predation and size do matter. It only means that much more variation exists among tanks for other reasons. As always, prediction and inference are just different tasks.

13M3. Now we want a slightly modified version of model `m1.1` from problem **13M1**. We just replace the Gaussian adaptive prior with a similar Cauchy prior. The

R code 13.7

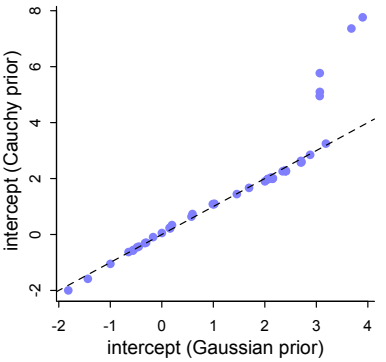
```
m1.1c <- ulam(
    alist(
        S ~ binomial( n , p ),
        logit(p) <- a[tank],
        a[tank] ~ dcauchy( a_bar , sigma ),
        a_bar ~ normal( 0 , 1 ),
        sigma ~ exponential( 1 )
    ), data=dat , chains=4 , cores=4 ,
    log_lik=TRUE , control=list(adapt_delta=0.99) )
```

You might have some trouble sampling efficiently from this posterior, on account of the long tails of the Cauchy. These result in the intercepts being poorly identified. You saw a simple example of this problem in Chapter 9, when you met MCMC and learned about diagnosing bad chains. This topic will come up in more detail in Chapter 13. In any event, be sure to check the chains carefully and sample more if you need to.

The problem asked you to compare the posterior means of the a parameters. Plotting the posterior means will be a lot more meaningful than just looking at the values.

R code 13.8

```
post1 <- extract.samples(m1.1)
a_tank1 <- apply(post1$a,2,mean)
post2 <- extract.samples(m1.1c)
a_tank2 <- apply(post2$a,2,mean)
plot( a_tank1 , a_tank2 , pch=16 , col=rangi2 ,
    xlab="intercept (Gaussian prior)" , ylab=" intercept (Cauchy prior)" )
abline(a=0,b=1,lty=2)
```



The dashed line show the values for which the intercepts are equal in the two models. You can see that for the majority of tank intercepts, the Cauchy model actually produces posterior means that are essentially the same as those from the Gaussian model. But the extremely large intercepts, under the Gaussian prior, are very much more extreme under the Cauchy prior. For those tanks, on the righthand side of the plot, all of the tadpoles survived. So using only the data from each tank alone, the log-odds of survival are infinite. The adaptive prior applies pooling that shrinks those log-odds inwards from infinity, thankfully. But the Gaussian prior causes more shrinkage of the extreme values than the Cauchy prior does. That is what accounts for those 5 extreme points on the right of the plot above.

13M4. To sample this posterior efficiently, we'll want to us a non-centered parameterization. The centered parameterization will work, but Stan will complain a lot. I'll use `transpars` to include the centered intercepts in the posterior:

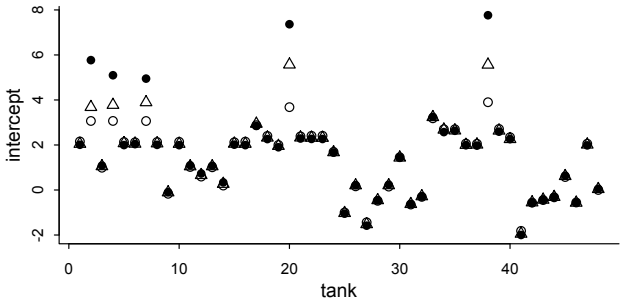
R code 13.9

```
m1.1t <- ulam(
    alist(
        S ~ binomial( n , p ),
        logit(p) <- a[tank],
        transpars> vector[tank]:a <<- a_bar + z*sigma,
        z[tank] ~ dstudent( 2 , 0 , 1 ),
        a_bar ~ normal( 0 , 1 ),
        sigma ~ exponential( 1 )
    ), data=dat , chains=4 , cores=4 ,
    log_lik=TRUE , control=list(adapt_delta=0.99) )
```

To compare all three posterior distributions:

R code 13.10

```
postn <- extract.samples( m1.1 )
an <- colMeans( postn$a )
postc <- extract.samples( m1.1c )
ac <- colMeans( postc$a )
postt <- extract.samples( m1.1t )
at <- colMeans( postt$a )
plot( NULL , xlim=c(1,48) , ylim=range(c(an,ac,at)) ,
    xlab="tank" , ylab="intercept" )
points( 1:48 , an )
points( 1:48 , ac , pch=16 )
points( 1:48 , at , pch=2 )
```



In most tanks, the posterior means are essentially the same. There are 5 tanks however where they differ. In each of these, there is the same ordering: Cauchy (filled) is most extreme, followed by Student-t (triangle), and then by Gaussian (open circle). This results from Cauchy having the thickest tails, followed by Student-t and then Gaussian. Thicker tails produce less shrinkage.

13M5. This is much like the model in the chapter, just with the two varying intercept means inside the two priors, instead of one mean outside both priors (inside the linear model). Since there are two parameters for the means, one inside each adaptive prior, this model is over-parameterized: an infinite number of different values of α and γ will produce the same sum $\alpha + \gamma$. The parameter γ is redundant, in other words. This will produce a poorly-identified posterior. It's best to avoid specifying a model like this. Now you'll see why.

Here's the code to prepare the data and fit the model:

R code 13.11

```
library(rethinking)
data(chimpanzees)
d <- chimpanzees
d$treatment <- 1 + d$prosoc_left + 2*d$condition
dat_list <- list(
    pulled_left = d$pulled_left,
    actor = d$actor,
    block_id = d$block,
    treatment = as.integer(d$treatment) )

m13M5 <- ulam(
    alist(
        pulled_left ~ dbinom( 1 , p ) ,
        logit(p) <- a[actor] + g[block_id] + b[treatment] ,
```

```
        b[treatment] ~ dnorm( 0 , 0.5 ),
      ## adaptive priors
        a[actor] ~ dnorm( a_bar , sigma_a ),
        g[block_id] ~ dnorm( g_bar , sigma_g ),
      ## hyper-priors
        a_bar ~ dnorm( 0 , 1.5 ),
        g_bar ~ dnorm( 0 , 1.5 ),
        sigma_a ~ dexp(1),
        sigma_g ~ dexp(1)
    ) , data=dat_list , chains=4 , cores=4 , log_lik=TRUE )
```

And just to make life easier, here's the code to re-fit the model from the chapter:

R code 13.12

```
m13.4 <- ulam(
    alist(
        pulled_left ~ dbinom( 1 , p ) ,
        logit(p) <- a[actor] + g[block_id] + b[treatment] ,
        b[treatment] ~ dnorm( 0 , 0.5 ),
      ## adaptive priors
        a[actor] ~ dnorm( a_bar , sigma_a ),
        g[block_id] ~ dnorm( 0 , sigma_g ),
      ## hyper-priors
        a_bar ~ dnorm( 0 , 1.5 ),
        sigma_a ~ dexp(1),
        sigma_g ~ dexp(1)
    ) , data=dat_list , chains=4 , cores=4 , log_lik=TRUE )
```

Now lets look at the `precis` output of each model:

R code 13.13

```
precis(m13.4 , 2 , pars=c("a_bar","b") )
precis(m13M5 , 2 , pars=c("a_bar","b","g_bar") )
```

```
       mean   sd  5.5% 94.5% n_eff Rhat4
a_bar  0.61 0.72 -0.58  1.75   949  1.01
b[1]  -0.15 0.32 -0.65  0.36   449  1.01
b[2]   0.38 0.32 -0.14  0.89   445  1.01
b[3]  -0.49 0.32 -1.00  0.01   440  1.01
b[4]   0.26 0.31 -0.22  0.77   439  1.01

       mean   sd  5.5% 94.5% n_eff Rhat4
a_bar  0.62 1.23 -1.35  2.55    22  1.11
b[1]  -0.16 0.29 -0.58  0.32   160  1.03
b[2]   0.39 0.29 -0.06  0.88   394  1.02
b[3]  -0.48 0.30 -0.96  0.02   492  1.01
b[4]   0.24 0.31 -0.21  0.75    40  1.07
g_bar  0.23 1.06 -1.34  2.01   150  1.02
```

The new model, `m13M5`, samples quite poorly. The `n_eff` values are much lower, and the `Rhat` values are larger. You may also have noticed that it samples slowly. This is what happens when you over-parameterize the intercept. Notice however that the inferences about the slopes are practically identical. So even though the over-parameterized model is inefficient, it has identified the slope parameters.

13M6. We can compute the posterior distributions with 4 simple `ulam` models:

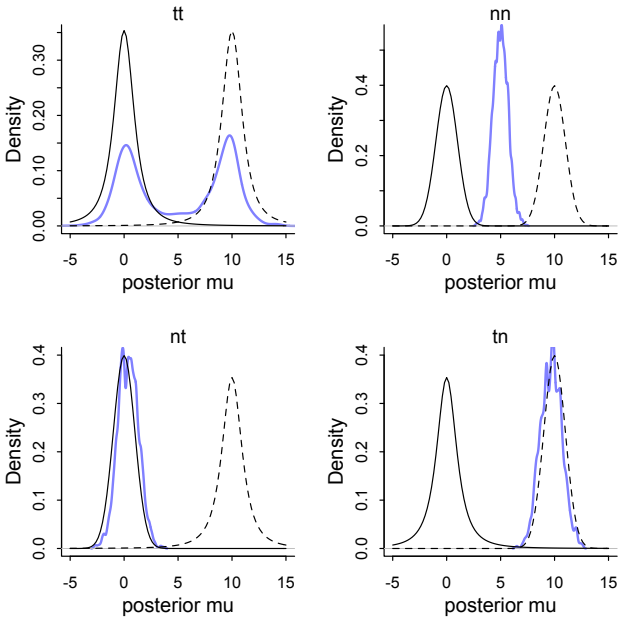
R code 13.14

```
mtt <- ulam(
    alist(
        y ~ dstudent(2,mu,1),
        mu ~ dstudent(2,10,1)
    ), data=list(y=0) , chains=4 )

mnn <- ulam(
    alist(
        y ~ dnorm(mu,1),
        mu ~ dnorm(10,1)
    ), data=list(y=0) , chains=4 )

mtn <- ulam(
    alist(
        y ~ dstudent(2,mu,1),
        mu ~ dnorm(10,1)
    ), data=list(y=0) , chains=4 )

mnt <- ulam(
    alist(
        y ~ dnorm(mu,1),
        mu ~ dstudent(2,10,1)
    ), data=list(y=0) , chains=4 )
```

Now to plot each posterior (blue) against the corresponding likelihoods (solid black) and priors (dashed):

R code 13.15

```
par(mfrow=c(2,2),cex=1.05)

p <- extract.samples(mtt)
dens(p$mu , xlim=c(-5,15) , ylim=c(0,0.35) , lwd=2 , col=rangi2 , xlab="posterior mu" )
mtext("tt")
curve( dstudent(0,2,x,1) , add=TRUE , lty=1 ) # lik
curve( dstudent(x,2,10,1) , add=TRUE , lty=2 ) # prior

p <- extract.samples(mnn)
dens(p$mu, xlim=c(-5,15), ylim=c(0,0.55), lwd=2 , col=rangi2, xlab="posterior mu")
mtext("nn")
curve( dnorm(0,x,1) , add=TRUE , lty=1 ) # lik
curve( dnorm(x,10,1) , add=TRUE , lty=2 ) # prior

p <- extract.samples(mnt)
dens(p$mu, xlim=c(-5,15), ylim=c(0,0.4), lwd=2 , col=rangi2, xlab="posterior mu")
mtext("nt")
curve( dnorm(0,x,1) , add=TRUE , lty=1 ) # lik
curve( dstudent(x,2,10,1) , add=TRUE , lty=2 ) # prior

p <- extract.samples(mtn)
dens(p$mu, xlim=c(-5,15), ylim=c(0,0.4), lwd=2 , col=rangi2, xlab="posterior mu")
mtext("tn")
curve( dstudent(0,2,x,1) , add=TRUE , lty=1 ) # lik
curve( dnorm(x,10,1) , add=TRUE , lty=2 ) # prior
```



The tt model is perhaps the most surprising—it has two peaks, one at the likelihood peak and another at the prior peak. The other three models have one posterior mode, but in different places. The nn model is a compromise between the likelihood and prior. The nt model prefers the likelihood and the tn model prefers the prior.

The explanation for this pattern is that the Student-t distribution, with its thick tails, does not aggressively pull posterior mass towards itself like the Gaussian, with its thin tails, does. So with both a Student-t likelihood and prior, neither dominates, and we end up with two modes. With two Gaussians, they tug each other to the middle. When either the likelihood or the prior is Student-t, the Student-t loses to the Gaussian.

There are two important lessons here. The first is that it is not easy to guess how prior and likelihood combine to form the posterior. Things like thickness of tails matter. The second is that we can use this fact to control what happens when the data (likelihood) and prior are incompatible. If we use thick-tailed priors, then the data will dominate. If instead we trust the prior more than the data (because of for example poor data quality), we might instead want to use a thick-tailed likelihood and thin-tailed prior. With enough data, the likelihood will still dominate.

13H1. Loading the data and prepping the data list:

R code 13.16

```
library(rethinking)
data(bangladesh)
d <- bangladesh
d$district_id <- as.integer(as.factor(d$district))

dat_list <- list(
    C = d$use.contraception,
    did = d$district_id )
```

Now for the ordinary fixed effect model:

R code 13.17

```
m13H1.1 <- ulam(
    alist(
        C ~ bernoulli( p ),
        logit(p) <- a[did],
        a[did] ~ normal( 0 , 1.5 )
    ) , data=dat_list , chains=4 , cores=4 , log_lik=TRUE )
```

And the varying intercepts model:

R code 13.18

```
m13H1.2 <- ulam(
    alist(
        C ~ bernoulli( p ),
        logit(p) <- a[did],
        a[did] ~ normal( a_bar , sigma ),
        a_bar ~ normal( 0 , 1.5 ),
        sigma ~ exponential( 1 )
    ) , data=dat_list , chains=4 , cores=4 , log_lik=TRUE )
```

Now let's extract the samples, compute posterior mean probabilities in each district, and plot it all. I'll plot fixed effects in blue, varying effects as open circles, and the raw empirical proportion in each district as a plus symbol.

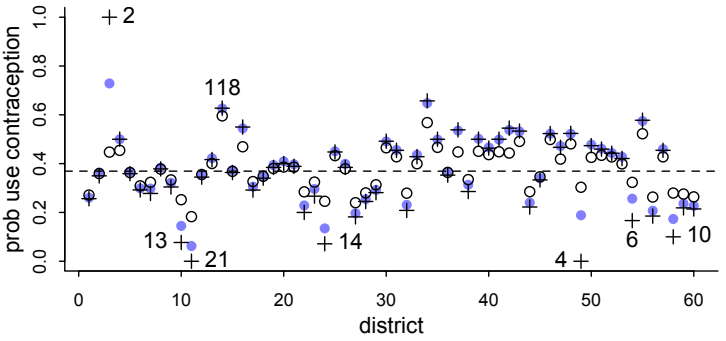
R code 13.19

```
post1 <- extract.samples( m13H1.1 )
post2 <- extract.samples( m13H1.2 )

p1 <- apply( inv_logit(post1$a) , 2 , mean )
p2 <- apply( inv_logit(post2$a) , 2 , mean )

# compute raw estimate from data in each district
t3 <- table( d$use.contraception , d$district_id )
n_per_district <- colSums( t3 )
p_raw <- as.numeric( t3[2,]/n_per_district )

nd <- max(dat_list$did)
plot( NULL , xlim=c(1,nd) , ylim=c(0,1) , ylab="prob use contraception" ,
    xlab="district" )
points( 1:nd , p1 , pch=16 , col=rangi2 )
points( 1:nd , p2 )
points( 1:nd , p_raw , pch=3 )
abline( h=mean(inv_logit(post2$a_bar)) , lty=2 )
```

Now I'll label a few points with the sample size in the district:

R code 13.20

```
identify( 1:nd , p_raw , labels=n_per_district )
```



As you'd expect, the varying intercepts (open circles) are shrunk towards the mean (the dashed line) relative to both the fixed intercepts (blue circles) and the raw proportions (plus symbols). Some are shrunk more than others. The third district from the left shrunk a lot. Let's look at the sample size in each district:

R code 13.21

```
table(d$district_id)
```

```
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
117  20   2  30  39  65  18  37  23  13  21  29  24 118  22  20  24  47  26  15

 21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
 18  20  15  14  67  13  44  49  32  61  33  24  14  35  48  17  13  14  26  41

 41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
 26  11  45  27  39  86  15  42   4  19  37  61  19   6  45  27  33  10  32  42
```

District 3 has only 2 women sampled. So it shrinks a lot. There are couple of other districts, like 49 and 54, that also have very few women sampled. But their fixed estimates aren't as extreme, so they don't shrink as much as district 3 does.

All of this is explained by partial pooling, of course.

13H2. First, let's load the data and re-run the old model from Chapter 12:

R code 13.22

```
data(Trolley)
d <- Trolley

dat <- list(
    R = d$response,
    A = d$action,
    I = d$intention,
    C = d$contact )

m13H2.1 <- ulam(
    alist(
        R ~ dordlogit( phi , cutpoints ),
        phi <- bA*A + bC*C + BI*I ,
        BI <- bI + bIA*A + bIC*C ,
        c(bA,bI,bC,bIA,bIC) ~ dnorm( 0 , 0.5 ),
        cutpoints ~ dnorm( 0 , 1.5 )
    ) , data=dat , chains=4 , cores=4 , log_lik=TRUE )
```

Now to run the varying intercept model, we need to build a valid individual ID variable. The IDs in the data are long tags, so we can coerce them to integers in many ways. What is important is that the index values go from 1 to the number of individuals.

R code 13.23

```
dat$id <- coerce_index( d$id )
```

Now we can run the model. The only additions here are the `a[id]` in the linear model and the adaptive prior for it. But I'll show the code for both the centered and non-centered parameterizations. The non-centered version should sample better. But both work.

R code 13.24

```
m13H2.2 <- ulam(
    alist(
        R ~ dordlogit( phi , cutpoints ),
        phi <- a[id] + bA*A + bC*C + BI*I ,
        BI <- bI + bIA*A + bIC*C ,
        a[id] ~ normal( 0 , sigma ),
        c(bA,bI,bC,bIA,bIC) ~ dnorm( 0 , 0.5 ),
        cutpoints ~ dnorm( 0 , 1.5 ),
        sigma ~ exponential(1)
    ) , data=dat , chains=4 , cores=4 , log_lik=TRUE )

m13H2.2z <- ulam(
    alist(
        R ~ dordlogit( phi , cutpoints ),
        phi <- z[id]*sigma + bA*A + bC*C + BI*I ,
        BI <- bI + bIA*A + bIC*C ,
        z[id] ~ normal( 0 , 1 ),
        c(bA,bI,bC,bIA,bIC) ~ dnorm( 0 , 0.5 ),
        cutpoints ~ dnorm( 0 , 1.5 ),
        sigma ~ exponential(1)
    ) , data=dat , chains=4 , cores=4 , log_lik=TRUE )
```

We can begin by comparing the posterior distributions. The original coefficients are:

R code 13.25

```
precis(m13H2.1)
```

```
     mean   sd  5.5% 94.5% n_eff Rhat4
bIC -1.24 0.10 -1.39 -1.07   997     1
bIA -0.43 0.08 -0.56 -0.31  1022     1
bC  -0.34 0.07 -0.45 -0.23   902     1
bI  -0.29 0.05 -0.38 -0.20   817     1
bA  -0.47 0.05 -0.55 -0.39  1013     1
```

And the new ones, having added the individual IDs, are:

R code 13.26

```
precis(m13H2.2z)
```

```
       mean   sd  5.5% 94.5% n_eff Rhat4
bIC   -1.67 0.10 -1.82 -1.51  1007  1.00
bIA   -0.56 0.08 -0.69 -0.43   870  1.00
bC    -0.45 0.07 -0.57 -0.34   927  1.00
bI    -0.39 0.06 -0.48 -0.29   837  1.00
```

```
bA    -0.65 0.06 -0.74 -0.56   840  1.00
sigma  1.90 0.08  1.78  2.04   159  1.01
```

Everything has gotten more negative. This is because there is a lot of individual variation in average rating—look at the distribution for `sigma`. That is on the logit scale, so that's a lot of variation on the probability scale. That variation in average rating was hiding some of the effect of the treatments. We get more precision by conditioning on individual.

The WAIC comparison can also help show how much variation comes from individual differences in average rating:

R code 13.27

```
compare( m13H2.1 , m13H2.2z )
```

```
           WAIC     SE  dWAIC    dSE pWAIC weight
m13H2.2z 31055.5 179.35    0.0     NA 355.1      1
m13H2.1  36928.4  80.70 5872.9 173.49  10.5      0
```

The WAIC difference is massive. This is consistent with individual variation in average rating being a major effect in this sample.

This is all quite typical of likert-scale data, in my experience. Individuals anchor on different points and this adds noise. When we have repeat samples from the same individual, we can condition away some of that noise and get more precise estimates of the treatment effects.

13H3. The cross-classified model will add additional varying intercepts for each `story` in the data. There are 12 different stories, which are repeated across individuals. So we have repeat measures on `story` just as we have repeat measures on individual `id`.

So let's load the data again and build the index variable for `story`:

R code 13.28

```
library(rethinking)
data(Trolley)
d <- Trolley
dat <- list(
    R = d$response,
    A = d$action,
    I = d$intention,
    C = d$contact,
    Sid = coerce_index(d$story),
    id = coerce_index(d$id) )
```

The cross-classified model just needs another set of varying intercepts clustered on story. Let's try these with a centered parameterization first. I'll call the story intercepts `s[Sid]` and their standard deviation `tau`. I'll also show the code for the non-centered version, which actually samples more efficiently. Here's the code:

R code 13.29

```
m13H3 <- ulam(
    alist(
        R ~ dordlogit( phi , cutpoints ),
        phi <- z[id]*sigma + s[Sid] + bA*A + bC*C + BI*I ,
        BI <- bI + bIA*A + bIC*C ,
        z[id] ~ normal( 0 , 1 ),
        s[Sid] ~ normal( 0 , tau ),
        c(bA,bI,bC,bIA,bIC) ~ dnorm( 0 , 0.5 ),
        cutpoints ~ dnorm( 0 , 1.5 ),
        sigma ~ exponential(1),
```

```
        tau ~ exponential(1)
    ) , data=dat , chains=4 , cores=4 , log_lik=TRUE )

m13H3z <- ulam(
    alist(
        R ~ dordlogit( phi , cutpoints ),
        phi <- z[id]*sigma + sz[Sid]*tau + bA*A + bC*C + BI*I ,
        BI <- bI + bIA*A + bIC*C ,
        z[id] ~ normal( 0 , 1 ),
        sz[Sid] ~ normal( 0 , 1 ),
        c(bA,bI,bC,bIA,bIC) ~ dnorm( 0 , 0.5 ),
        cutpoints ~ dnorm( 0 , 1.5 ),
        sigma ~ exponential(1),
        tau ~ exponential(1)
    ) , data=dat , chains=4 , cores=4 , log_lik=TRUE )
```

Let's look first at the marginal posterior distribution for `m13H3z`:

```
precis(m13H3z)
```

R code 13.30

```
       mean   sd  5.5% 94.5% n_eff Rhat4
bIC   -1.29 0.11 -1.47 -1.11  1721  1.00
bIA   -0.53 0.09 -0.67 -0.39  1708  1.00
bC    -1.08 0.10 -1.24 -0.92  1506  1.00
bI    -0.46 0.07 -0.57 -0.34  1697  1.00
bA    -0.89 0.07 -1.00 -0.78  1434  1.00
sigma  1.97 0.08  1.85  2.11   343  1.01
tau    0.55 0.15  0.38  0.80   396  1.01
```

The standard deviation among individuals, `sigma`, is similar to what it was in `m13H2z`. The posterior mean standard deviation among stories, `tau`, is about a third as large. So there's more variation among individuals than among stories.

Including varying intercepts on stories has always had a noticeable impact on estimates for the treatment variables. So again, variation across clusters in the presence of repeat measures plausibly biased the treatment estimate. But the qualitative story stays the same. This model improves precision, but it isn't telling a different causal story.

13H4. This problem is very similar to **13M1**, but it asks for interpretation of the posterior distribution. The same code is needed. Run the models from that solution again. Then let's inspect the posterior distributions of the coefficients. First the model with only predation:

```
precis( m1.2 )
```

R code 13.31

```
       mean   sd  5.5% 94.5% n_eff Rhat4
bp    -2.43 0.29 -2.90 -1.95   264  1.01
a_bar  2.53 0.23  2.17  2.91   286  1.00
sigma  0.82 0.14  0.61  1.07   694  1.00
```

Predation has a very strong negative effect on survival, which makes sense. Now consider the model that omits predation but includes size:

R code 13.32
```
precis( m1.3 , 2 , pars="s" )
```

```
      mean   sd  5.5% 94.5% n_eff Rhat4
s[1]  0.24 0.37 -0.35  0.82   237  1.01
s[2] -0.06 0.38 -0.64  0.55   239  1.01
```

Not such a clear effect of size. Now let's consider the models that include both. First the model without an interaction:

R code 13.33
```
precis( m1.4 , 2 , pars=c("bp","s") )
```

```
      mean   sd  5.5% 94.5% n_eff Rhat4
bp   -2.47 0.28 -2.92 -2.02   625  1.00
s[1]  0.32 0.37 -0.26  0.91   184  1.02
s[2] -0.11 0.37 -0.70  0.47   151  1.03
```

The agrees with the previous models. Predation has a clear and large impact. Size not so much. Now the interaction model:

R code 13.34
```
precis( m1.5 , 2 , pars=c("bp","s") )
```

```
       mean   sd  5.5% 94.5% n_eff Rhat4
bp[1] -1.86 0.38 -2.45 -1.24   906  1.01
bp[2] -2.77 0.38 -3.36 -2.17   899  1.00
s[1]   0.12 0.39 -0.52  0.73  1117  1.00
s[2]   0.16 0.40 -0.46  0.80  1295  1.00
```

The effect of predation does seem to vary by size. Let's compute the contrast:

R code 13.35
```
post <- extract.samples( m1.5 )
quantile( post$bp[,2] - post$bp[,1] , c(0.055,0.5,0.945) )
```

```
       5.5%         50%       94.5%
-1.68585188 -0.93005107 -0.08262948
```

So the contrast is reliably negative. Seems like size does matter, but only as it influences predation.