10.1.3. Aggregated binomial: Graduate school admissions.

Often the number of trials on each row is not a constant. So then in place of the "`18`" we insert a variable from the data. Let's work through an example. First, load the data:

R code 10.22

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

This data table only has 12 rows, so let's look at the entire thing:

```
   dept applicant.gender admit reject applications
1     A             male   512    313          825
2     A           female    89     19          108
3     B             male   353    207          560
4     B           female    17      8           25
5     C             male   120    205          325
6     C           female   202    391          593
7     D             male   138    279          417
8     D           female   131    244          375
9     E             male    53    138          191
10    E           female    94    299          393
11    F             male    22    351          373
12    F           female    24    317          341
```

These are graduate school applications to 6 different academic departments at UC Berkeley.[141] The `admit` column indicates the number offered admission. The `reject` column indicates the opposite decision. The `applications` column is just the sum of `admit` and `reject`. Each application has a 0 or 1 outcome for admission, but since these outcomes have been aggregated by department and gender, there are only 12 rows. These 12 rows however represent 4526 applications, the sum of the `applications` column. So there is a lot of data here—counting the rows in the data table is no longer a sensible way to assess sample size. We could split these data apart into 0/1 Bernoulli trials, like in the original `chimpanzees` data. Then there would be 4526 rows in the data.

Our job is to evaluate whether these data contain evidence of gender bias in admissions. We will model the admission decisions, focusing on applicant gender as a predictor variable. So we want to fit at least two models:

(1) A binomial regression that models `admit` as a function of each applicant's gender. This will estimate the association between gender and probability of admission.
(2) A binomial regression that models `admit` as a constant, ignoring gender. This will allow us to get a sense of any overfitting committed by the first model.

This is what the first model looks like, in mathematical form:

$$\begin{aligned}
n_{\text{admit},i} &\sim \text{Binomial}(n_i, p_i) \\
\text{logit}(p_i) &= \alpha + \beta_m m_i \\
\alpha &\sim \text{Normal}(0, 10) \\
\beta_m &\sim \text{Normal}(0, 10)
\end{aligned}$$

The variable n_i indicates `applications[i]`, the number of applications on row i. The predictor m_i is a dummy that indicates "male." We'll construct it just before fitting both models, like this:

R code 10.23

```
d$male <- ifelse( d$applicant.gender=="male" , 1 , 0 )
m10.6 <- map(
    alist(
        admit ~ dbinom( applications , p ) ,
        logit(p) <- a + bm*male ,
        a ~ dnorm(0,10) ,
        bm ~ dnorm(0,10)
    ) ,
    data=d )
m10.7 <- map(
    alist(
        admit ~ dbinom( applications , p ) ,
        logit(p) <- a ,
        a ~ dnorm(0,10)
    ) ,
    data=d )
```

A quick WAIC comparison verifies that the `male` predictor variable improves expected out-of-sample deviance by a very large amount:

R code 10.24

```
compare( m10.6 , m10.7 )
```

```
        WAIC pWAIC dWAIC weight    SE   dSE
m10.6 5954.9     2   0.0      1 34.98    NA
m10.7 6046.3     1  91.5      0 29.93 19.13
```

This comparison suggests that gender matters a lot. To see how it matters, we have to look at estimates for m10.6:

R code 10.25

```
precis(m10.6)
```

```
    Mean StdDev  5.5% 94.5%
a  -0.83   0.05 -0.91 -0.75
bm  0.61   0.06  0.51  0.71
```

Seems like being male is an advantage in this context. You can compute the relative difference in admission odds as $\exp(0.61) \approx 1.84$. This means that a male applicant's odds were 184% of a female applicant's. On the absolute scale, which is what matters, the difference in probability of admission is:

R code 10.26

```
post <- extract.samples( m10.6 )
p.admit.male <- logistic( post$a + post$bm )
p.admit.female <- logistic( post$a )
diff.admit <- p.admit.male - p.admit.female
quantile( diff.admit , c(0.025,0.5,0.975) )
```

```
     2.5%       50%     97.5%
0.1132778 0.1413527 0.1693274
```

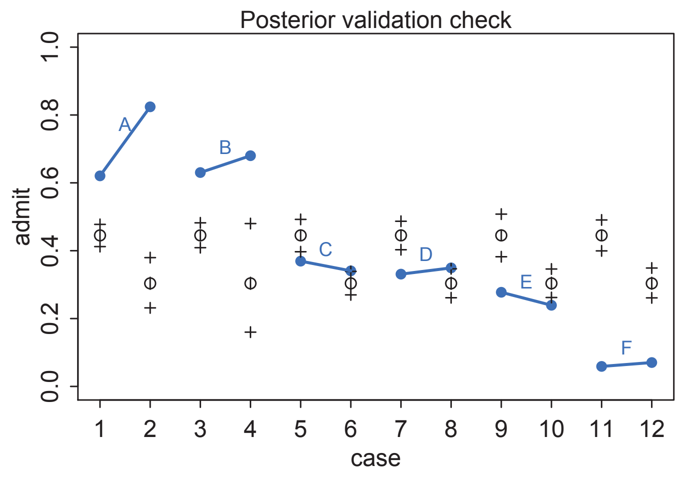

FIGURE 10.5. Posterior validation for model `m10.6`. Blue points are observed proportions admitted for each row in the data, with points from the same department connected by a blue line. Open points, the tiny vertical black lines within them, and the crosses are expected proportions, 89% intervals of the expectation, and 89% interval of simulated samples, respectively.

This means that the median estimate of the male advantage is about 14%, with a 95% interval from 11% to almost 17%. You may also want to inspect the density plot: `dens(diff.admit)` (not shown).

Before moving on to speculate on the cause of the male advantage, let's plot posterior predictions for the model. We'll use the default posterior validation check function, `postcheck`, and then dress it up a little by adding lines to connect data points from the same department.

R code
10.27
```
postcheck( m10.6 , n=1e4 )
# draw lines connecting points from same dept
for ( i in 1:6 ) {
    x <- 1 + 2*(i-1)
    y1 <- d$admit[x]/d$applications[x]
    y2 <- d$admit[x+1]/d$applications[x+1]
    lines( c(x,x+1) , c(y1,y2) , col=rangi2 , lwd=2 )
    text( x+0.5 , (y1+y2)/2 + 0.05 , d$dept[x] , cex=0.8 , col=rangi2 )
}
```

The result is shown as FIGURE 10.5. Those are pretty terrible predictions. There are only two departments in which females had a lower rate of admission than males (C and E), and yet the model says that females should expect to have a 14% lower chance of admission.

Sometimes a fit this bad is the result of a coding mistake. In this case, it is not. The model did correctly answer the question we asked of it: *What are the average probabilities of admission for females and males, across all departments?* The problem in this case is that males and females do not apply to the same departments, and departments vary in their rates of admission. This makes the answer misleading. You can see the steady decline in admission probability for both males and females from department A to department F. Females in these data tended not to apply to departments like A and B, which had high overall admission rates. Instead they applied in large numbers to departments like F, which admitted less than 10% of applicants.

So while it is true overall that females had a lower probability of admission in these data, it is clearly not true within most departments. And note that just inspecting the posterior distribution alone would never have revealed that fact to us. We had to appeal to something outside the fit model. In this case, it was a simple posterior validation check.

Instead of asking "*What are the average probabilities of admission for females and males across all departments?*" we want to ask "*What is the average difference in probability of admission between females and males within departments?*" In order to ask the second question, we estimate a unique female admission rate in each department—an intercept—and then an average male difference. Here's a model that asks this new question:

$$\begin{aligned} n_{\text{admit},i} &\sim \text{Binomial}(n_i, p_i) \\ \text{logit}(p_i) &= \alpha_{\text{DEPT}[i]} + \beta_m m_i \\ \alpha_{\text{DEPT}} &\sim \text{Normal}(0, 10) \\ \beta_m &\sim \text{Normal}(0, 10) \end{aligned}$$

where DEPT indexes department. So now each department gets its own log-odds of admission, α_{DEPT}, but the model still estimates a universal adjustment—same in all departments—for a male application, β_m.

Fitting this model, along with a version that omits `male`, is straightforward. We'll use the indexing notation again to construct an intercept for each department. But first, we also need to construct a numerical index that numbers the departments 1 through 6. The function `coerce_index` can do this for us, using the `dept` factor as input. Here's the code to construct the index and fit both models:

R code 10.28

```
# make index
d$dept_id <- coerce_index( d$dept )

# model with unique intercept for each dept
m10.8 <- map(
    alist(
        admit ~ dbinom( applications , p ) ,
        logit(p) <- a[dept_id] ,
        a[dept_id] ~ dnorm(0,10)
    ) , data=d )

# model with male difference as well
m10.9 <- map(
    alist(
        admit ~ dbinom( applications , p ) ,
        logit(p) <- a[dept_id] + bm*male ,
        a[dept_id] ~ dnorm(0,10) ,
        bm ~ dnorm(0,10)
    ) , data=d )
```

Now to compare all four models from this section:

R code 10.29
```
compare( m10.6 , m10.7 , m10.8 , m10.9 )
```

```
        WAIC pWAIC dWAIC weight    SE   dSE
m10.8 5200.9     6   0.0   0.56 57.02    NA
m10.9 5201.4     7   0.5   0.44 57.06  2.48
m10.6 5954.8     2 753.9   0.00 34.98 48.53
m10.7 6046.3     1 845.4   0.00 29.95 52.37
```

The new models fit much better, unsurprisingly. But now the model without `male` is ranked first. Still, the WAIC difference between `m10.8` and `m10.9` is tiny—both models get about half the Akaike weight. I'd call this a tie. So there's modest support for some effect of gender, even if it is overfit a little. So let's look at the estimates from m10.9 and see how the estimated association of gender with admission has changed:

R code 10.30
```
precis( m10.9 , depth=2 )
```

```
      Mean StdDev  5.5% 94.5%
a[1]  0.68   0.10  0.52  0.84
a[2]  0.64   0.12  0.45  0.82
a[3] -0.58   0.07 -0.70 -0.46
a[4] -0.61   0.09 -0.75 -0.48
a[5] -1.06   0.10 -1.22 -0.90
a[6] -2.62   0.16 -2.88 -2.37
bm   -0.10   0.08 -0.23  0.03
```

The estimate for `bm` goes in the opposite direction now. On the proportional odds scale, the estimate becomes: $\exp(-0.1) \approx 0.9$. So a male in this sample has about 90% the odds of admission as a female, comparing within departments.

We can also plot posterior predictions again and see how the department intercepts capture the variation in overall admission rates. You can use the same code as earlier, just replacing `m10.6` inside the call to `postcheck` with `m10.9`. The result is shown as FIGURE 10.6.

Before moving on, let's check the quadratic approximation. Since it turned out in the `chimpanzees` data that individual intercepts caused problems for quadratic approximation, it's worth checking here as well. To fit model `m10.9` with MCMC:

R code 10.31
```
m10.9stan <- map2stan( m10.9 , chains=2 , iter=2500 , warmup=500 )
precis(m10.9stan,depth=2)
```

```
      Mean StdDev lower 0.89 upper 0.89 n_eff Rhat
a[1]  0.68   0.10       0.52       0.84  1463    1
a[2]  0.64   0.12       0.45       0.82  1510    1
a[3] -0.58   0.07      -0.70      -0.47  2267    1
a[4] -0.61   0.09      -0.75      -0.48  2166    1
a[5] -1.06   0.10      -1.22      -0.91  2755    1
a[6] -2.64   0.17      -2.89      -2.36  3012    1
bm   -0.10   0.08      -0.23       0.03  1224    1
```

These estimates are practically identical to those from `m10.9`. Go ahead and inspect the `pairs` plot, too. You'll see that the quadratic approximation has done a very good job. This

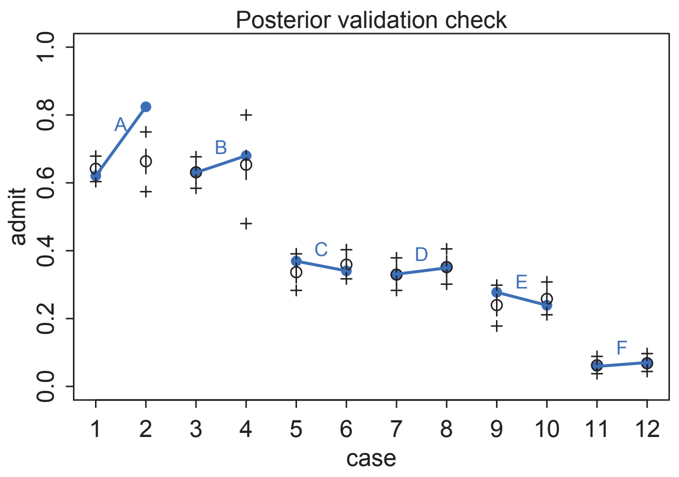

FIGURE 10.6. Posterior validation for `m10.9`. The unique intercepts for each department, A through F, capture variation in overall admission rates among departments. This allows the model to compare male and female admission rates, controlling for heterogeneity across departments.

is typical for binomial regression, as long as none of the intercepts are very far from zero, such that they push against the ceiling or floor, and none of the predictors are strongly associated with the outcome. Then the quadratic approximation can be very accurate. But since it's easy to check, it's usually worth a check.

Rethinking: Simpson's paradox is not a paradox. This empirical example is a famous one in statistical teaching. It is often used to illustrate a phenomenon known as **SIMPSON'S PARADOX**.[142] Like most paradoxes, there is no violation of logic, just of intuition. And since different people have different intuition, Simpson's paradox means different things to different people. The poor intuition being violated in this case is that a positive association in the entire population should also hold within each department. Overall, females in these data did have a harder time getting admitted to graduate school. But that arose because females applied to the hardest departments for anyone, male or female, to gain admission to.

Perhaps a little more paradoxical is that this phenomenon can repeat itself indefinitely within a sample. Any association between an outcome and a predictor can be nullified or reversed when another predictor is added to the model. All that we can do about this is to remain skeptical of models and try to imagine ways they might be deceiving us. Thinking causally about these settings sometimes helps.[143] But also see the box about causal inference on page 120.

Overthinking: WAIC and aggregated binomial models. The `WAIC` function in `rethinking` detects aggregated binomial models and automatically splits them apart into 0/1 Bernoulli trials, for the purpose of calculating WAIC. It does this, because WAIC is computed point by point (see Chapter 6). So what you define as a "point" affects WAIC's value. In an aggregated binomial each "point" is a bunch of independent trials that happen to share the same predictor values. In order for the disaggregated and aggregated models to agree, it makes sense to use the disaggregated representation.

A consequence of this is that DIC and WAIC for an aggregated binomial will look very different from one another, even though they will produce similar model rankings. Try for example comparing `compare(m10.8,m10.9,func=DIC)` to the WAIC table for the same two models. The reason the absolute magnitudes of WAIC and DIC end up so different in these cases is because the binomial distribution, in aggregated form, has a leading coefficient that is the *multiplicity*, the number of ways that the sequence of 0/1 outcomes could be reordered. This coefficient doesn't change inference, since it isn't a function of the parameters. But it does change the value of the log-likelihood and therefore the deviance.

10.1.4. Fitting binomial regressions with `glm`. R's standard `glm` function allows fitting a variety of generalized linear models, as long as you are okay with and cautious of flat priors. An aggregated binomial uses `cbind` to build the outcome variable. For example, this code will yield similar results as the `map` approach in the previous section:

R code 10.32
```
m10.7glm <- glm( cbind(admit,reject) ~ 1 , data=d , family=binomial )
m10.6glm <- glm( cbind(admit,reject) ~ male , data=d , family=binomial )
m10.8glm <- glm( cbind(admit,reject) ~ dept , data=d , family=binomial )
m10.9glm <- glm( cbind(admit,reject) ~ male + dept , data=d ,
    family=binomial )
```

When the outcome is instead coded as 0/1, the input looks like a linear regression formula:

R code 10.33
```
data(chimpanzees)
m10.4glm <- glm(
    pulled_left ~ as.factor(actor) + prosoc_left * condition - condition ,
    data=chimpanzees , family=binomial )
```

Note the necessity of subtracting `condition` to remove that main effect from the model.

And you can use `glimmer` to build a `map`-style model from one of these linear model formulas. For example:

R code 10.34
```
glimmer( pulled_left ~ prosoc_left * condition - condition ,
    data=chimpanzees , family=binomial )
```

```
alist(
    pulled_left ~ dbinom( 1 , p ),
    logit(p) <- Intercept +
        b_prosoc_left*prosoc_left +
        b_prosoc_left_X_condition*prosoc_left_X_condition,
    Intercept ~ dnorm(0,10),
    b_prosoc_left ~ dnorm(0,10),
    b_prosoc_left_X_condition ~ dnorm(0,10)
)
```

The parameter names are inelegant, but you can edit the above to your liking.

Notice that `glimmer` above inserts weakly regularizing priors by default (see `?glimmer` for options). Sometimes, the implicit flat priors of `glm` lead to nonsense estimates. For example, consider the following simple data and model context:

R code 10.35
```
# outcome and predictor almost perfectly associated
y <- c( rep(0,10) , rep(1,10) )
```