12 Monsters and Mixtures

In Hawaiian legend, Nanaue was the son of a shark who fell in love with a human women. He grew into a murderous man with a shark mouth in the middle of his back. In Greek legend, the minotaur was a man with the head of a bull. He was the spawn of a human woman and a great white bull. The gryphon is a legendary monster that is part eagle and part lion. Maori legends speak of *Taniwha*, monsters with features of serpents and birds and even sharks, much like the dragons of Chinese and European mythology.

By piecing together parts of different creatures, it's easy to make a monster. Many monsters are hybrids. Many statistical models are too. This chapter is about constructing likelihood and link functions by piecing together the simpler components of previous chapters. Like legendary monsters, these hybrid likelihoods contain pieces of other model types. Endowed with some properties of each piece, they help us model outcome variables with inconvenient, but common, properties. Being monsters, these models are both powerful and dangerous. They are often harder to estimate and to understand. But with some knowledge and caution, they are important tools.

We'll consider three common and useful examples. The first are models for handling **over-dispersion**. These models extend the binomial and Poisson models of the previous chapter to cope a bit with unmeasured sources of variation. The second type is a family of **zero-inflated** and **zero-augmented** models, each of which mixes a binary event with an ordinary GLM likelihood like a Poisson or binomial. The third type is the **ordered categorical** model, useful for categorical outcomes with a fixed ordering. This model is built by merging a categorical likelihood function with a special kind of link function, usually a **cumulative link**. We'll also learn how to construct ordered categorical predictors.

These model types help us transform our modeling to cope with the inconvenient realities of measurement, rather than transforming measurements to cope with the constraints of our models. There are lots of other model types that arise for this purpose and in this way, by mixing bits of simpler models together. We can't possibly cover them all. But when you encounter a new type, at least you'll have a framework in which to understand it. And if you ever need to construct your own unique monster, feel free to do so. Just be sure to validate it by simulating dummy data and then recovering the data-generating process through fitting the model to the dummy data.

12.1. Over-dispersed counts

In an earlier chapter (Chapter 7), I argued that models based on normal distributions can be overly sensitive to extreme observations. The problem isn't necessary that "outliers" are bad data. Rather processes are often variable mixtures and this results in thicker tails.

Models that assume a thin tail, like a pure Gaussian model, can be easily excited. Using something like a Student's t instead can help produce better inferences and out-of-sample predictions.

The same goes for count models. When counts arise from a mixture of different processes, then there may be more variation—thicker tails—than a pure count model expects. This can again lead to overly excited models. When counts are more variable than a pure process, they exhibit **OVER-DISPERSION**. The variance of a variable is sometimes called its **DISPERSION**. For a counting process like a binomial, the variance is a function of the same parameters as the expected value. For example, the expected value of a binomial is Np and its variance is $Np(1 - p)$. When the observed variance exceeds this amount—after conditioning on all the predictor variables—this implies that some omitted variable is producing additional dispersion in the observed counts.

That isn't necessarily bad. Such a model could still produce perfectly good inferences. But ignoring over-dispersion can also lead to all of the same problems as ignoring any predictor variable. Heterogeneity in counts can be a confound, hiding effects of interest or producing spurious inferences. So it's worth trying grappling with over-dispersion. The best solution would of course be to discover the omitted source of dispersion and include it in the model. But even when no additional variables are available, it is possible to mitigate the effects of over-dispersion. We'll consider two common and useful strategies.

In this chapter, we'll consider **CONTINUOUS MIXTURE** models in which a linear model is attached not to the observations themselves but rather to a distribution of observations. We'll spend the rest of this section outlining this kind of model, using the common beta-binomial and gamma-Poisson (negative-binomial) models of this type. These models were mentioned at the end of the previous chapter, but now we'll actually define them.

In the next chapters, we'll see how to employ multilevel models that estimate both the residuals of each observation and the distribution of those residuals. In practice, it is often easier to use multilevel models (GLMMs, Chapter 13) in place of continuous mixtures. The reason is that multilevel models are much more flexible. They can handle over-dispersion and other kinds of heterogeneity at the same time.

12.1.1. Beta-binomial.

A **BETA-BINOMIAL** model assumes that each binomial count observation has its own probability of a success.[175] The model estimates the *distribution* of probabilities of success across cases, instead of a single probability of success. And predictor variables change the shape of this distribution, instead of directly determining the probability of each success.

This will be easier to understand in the context of an example. For example, the `UCBadmit` data that you met last chapter is quite over-dispersed, as long as we ignore department. This is because the departments vary a lot in baseline admission rates. You've already seen that ignoring this variation leads to an incorrect inference about applicant gender. Now let's fit a beta-binomial model, ignoring department, and see how it picks up on the variation that arises from the omitted variable.

What a beta-binomial model of these data will assume is that each observed count on each row of the data table has its own unique, unobserved probability of admission. These probabilities of admission themselves have a common distribution. This distribution is described using a beta distribution, which is a probability distribution for probabilities. Why use a beta distribution? Because it makes the mathematics easy. When we use a beta, it is mathematically possible to solve for a closed form likelihood function that averages over the

unknown probabilities for each observation. See the Overthinking box at the end of this section (page 387) for details.

A beta distribution has two parameters, an average probability $\bar{p}$ and a shape parameter θ.[176] The shape parameter θ describes how spread out the distribution is. When $\theta = 2$, every probability from zero to one is equally likely. As θ increases above 2, the distribution of probabilities grows more concentrated. When $\theta < 2$, the distribution is so dispersed that extreme probabilities near zero and one are more likely than the mean. You can play around with the parameters to get a feel for the shapes this distribution can take:

R code 12.1

```
pbar <- 0.5
theta <- 5
curve( dbeta2(x,pbar,theta) , from=0 , to=1 ,
    xlab="probability" , ylab="Density" )
```

Explore different values for `pbar` and `theta` in the code above. Remember, this is a distribution for probabilities, so the horizontal axis you'll see represents different possible probability values, and the vertical axis is the density with which each probability on the horizontal is sampled from the distribution. It's weird, but you'll get used to it.

We're going to bind our linear model to $\bar{p}$, so that changes in predictor variables change the central tendency of the distribution. In mathematical form, the model is:

$$\begin{aligned} A_i &\sim \text{BetaBinomial}(N_i, \bar{p}_i, \theta) \\ \text{logit}(\bar{p}_i) &= \alpha_{\text{GID}[i]} \\ \alpha_j &\sim \text{Normal}(0, 1.5) \\ \theta &= \phi + 2 \\ \phi &\sim \text{Exponential}(1) \end{aligned}$$

where the outcome A is `admit`, the size N is `applications`, and $\text{GID}[i]$ is gender id, 1 for male and 2 for female. I've introduced a trick with the prior on θ. We want to assume that the dispersion is at least 2, which means flat. Less than 2 would be piling up probability on 0 and 1. Greater than 2 is increasingly heaped on a single value. Which distribution has a minimum of 2? We can make one. The exponential has a minimum of zero. But if add 2 to any exponentially distribution variable, then the minimum of the new variable is 2. So the model above defines ϕ with an exponential distribution. And then

The code below will load the data and then fit, using `ulam`, the beta-binomial model:

R code 12.2

```
library(rethinking)
data(UCBadmit)
d <- UCBadmit
d$gid <- ifelse( d$applicant.gender=="male" , 1L , 2L )
dat <- list( A=d$admit , N=d$applications , gid=d$gid )
m12.1 <- ulam(
    alist(
        A ~ dbetabinom( N , pbar , theta ),
        logit(pbar) <- a[gid],
        a[gid] ~ dnorm( 0 , 1.5 ),
        transpars> theta <<- phi + 2.0,
        phi ~ dexp(1)
```

```
    ), data=dat , chains=4 )
```

I tagged `theta` with `transpars>` (transformed parameters) so that Stan will return it in the samples. Let's take a quick look at the posterior means. But let's also go ahead and compute the constrast between the two genders first:

R code 12.3

```
post <- extract.samples( m12.1 )
post$da <- post$a[,1] - post$a[,2]
precis( post , depth=2 )
```

```
ulam posterior: 2000 samples from m12.1
       mean   sd 5.5% 94.5%   histogram
a[1]  -0.45 0.41 -1.1  0.21       ▁▂▇▃▁
a[2]  -0.34 0.40 -1.0  0.27       ▁▂▅▂▁
phi    1.05 0.78  0.1  2.44 ▇▅▃▂▁▁▁▁▁▁
theta  3.05 0.78  2.1  4.44 ▇▅▃▂▁▁▁▁▁▁
da    -0.11 0.57 -1.0  0.76   ▁▂▅▇▃▂▁▁
```

The parameter `a[1]` is the log-odds of admission for male applicants. It is lower than `a[2]`, the log-odds for female applicants. But the difference between the two, `da`, is highly uncertain. There isn't much evidence here of a difference between male and female admission rates. Recall that in the previous chapter, a binomial model of these data that omitted department ended up misleading, because there is an indirect path from gender through department to admission. That confound resulted in a spurious indication that female applicants had lower odds of admission. But the model above is not confounded, despite not containing the department variable. How is this?

The beta-binomial model allows each row in the data—each combination of department and gender—to have its own unobserved intercept. These unobserved intercepts are sampled from a beta distribution with mean $\bar{p}_i$ and dispersion θ. To see what this beta distribution looks like, we can just plot it.

R code 12.4

```
gid <- 2
# draw posterior mean beta distribution
curve( dbeta2(x,mean(logistic(post$a[,gid])),mean(post$theta)) , from=0 , to=1 ,
    ylab="Density" , xlab="probability admit", ylim=c(0,3) , lwd=2 )

# draw 50 beta distributions sampled from posterior
for ( i in 1:50 ) {
    p <- logistic( post$a[i,gid] )
    theta <- post$theta[i]
    curve( dbeta2(x,p,theta) , add=TRUE , col=col.alpha("black",0.2) )
}
mtext( "distribution of female admission rates" )
```

The result is shown on the left in FIGURE 12.1. Remember that a posterior distribution simultaneously scores the plausibility of every combination of parameter values. This plot shows 50 combinations of $\bar{p}$ and θ, sampled from the posterior. The thick curve is the beta distribution corresponding the posterior mean. The central tendency is for low probabilities of admission, less than 0.5. But the most plausible distributions allow for departments

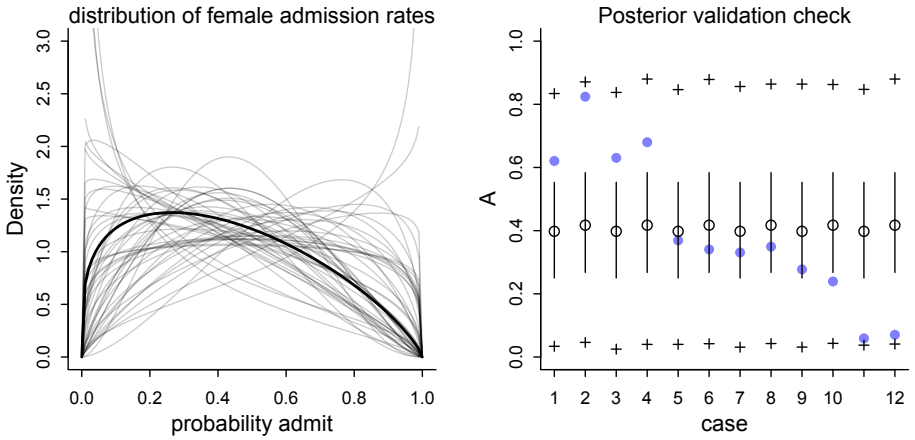

FIGURE 12.1. Left: Posterior distribution of beta distributions for `m12.1`. The thick curve is the posterior mean beta distribution. The lighter curves represent 100 combinations of $\bar{p}$ and θ sampled from the posterior. Right: Posterior validation check for `m12.1`. As a result of the widely dispersed beta distributions on the left, the raw data (blue) is contained within the prediction intervals.

that admit most applicants. What the model has done is accommodate the variation among departments—there is a lot of variation! As a result, it is no longer tricked by department variation into a false inference about gender.

To get a sense of how the beta distribution of probabilities of admission influences predicted counts of applications admitted, let's look at the posterior validation check:

```
postcheck( m12.1 )
```

R code 12.5

This plot is shown on the right in FIGURE 12.1. The vertical axis shows the predicted proportion admitted, for each case on the horizontal. The blue points show the empirical proportion admitted on each row of the data. The open circles are the posterior mean $\bar{p}$, with 89% percentile interval, and the + symbols mark the 89% interval of predicted counts of admission. There is a lot of dispersion expected here. The model can't see departments, because we didn't tell it about them. But it does see heterogeneity across rows, and it uses the beta distribution to estimate and anticipate that heterogeneity.

12.1.2. Negative-binomial or gamma-Poisson. A NEGATIVE-BINOMIAL model, more usefully called a GAMMA-POISSON model, assumes that each Poisson count observation has its own rate.[177] It estimates the shape of a gamma distribution to describe the Poisson rates across cases. Predictor variables adjust the shape of this distribution, not the expected value of each observation. The gamma-Poisson model is very much like a beta-binomial model, with the gamma distribution of rates (or expected values) replacing the beta distribution of probabilities of success. Why gamma? Because it makes the mathematics easy—there is a simple analytical expression for Poisson probabilities that are mixed together with gamma

distributed rates. These gamma-Poisson models are very useful. The reason is that Poisson distributions are very narrow. The variance must equal the mean, recall.

The gamma-Poisson distribution has two parameters, one for the mean (rate) and another for the dispersion (scale) of the rates across cases.

$$y_i \sim \text{Gamma-Poisson}(\lambda_i, \phi)$$

The λ parameter can be treated like the rate of an ordinary Poisson. The ϕ parameter must be positive and controls the variance. The variance of the gamma-Poisson is $\lambda + \lambda^2/\phi$. So larger ϕ values mean the distribution is more similar to a pure Poisson process.

Let's see how this works with the Oceanic tools example from the previous chapter. There was a highly influential point, Hawaii, that will become much less influential in the equivalent gamma-Poisson model. Why? Because gamma-Poisson expects more variation around the mean rate. As a result, Hawaii ends up pulling the regression trend less.

R code 12.6

```
library(rethinking)
data(Kline)
d <- Kline
d$P <- standardize( log(d$population) )
d$contact_id <- ifelse( d$contact=="high" , 2L , 1L )

dat2 <- list(
    T = d$total_tools,
    P = d$population,
    cid = d$contact_id )

m12.2 <- ulam(
    alist(
        T ~ dgampois( lambda , phi ),
        lambda <- exp(a[cid])*P^b[cid] / g,
        a[cid] ~ dnorm(1,1),
        b[cid] ~ dexp(1),
        g ~ dexp(1),
        phi ~ dexp(1)
    ), data=dat2 , chains=4 , log_lik=TRUE )
```

The posterior predictions of the new gamma-Poison model are displayed against the raw data in FIGURE 12.2. Model `m11.11` from the previous chapter, the pure Poisson model, is shown next to it for comparison. Recall that Hawaii was a highly influential point in the pure Poisson model. It does all the work of pulling the low-contact trend down. In this new model, Hawaii is still influential, but it exerts a lot less influence on the trends. Now the high and low contact trends are much more similar, very hard to reliably distinguish. This is because the gamma-Poisson model expects rate variation, and the estimated amount of variation is quite large. Population is still strongly related to the total tools, but the influence of contact rate has greatly diminished.

12.1.3. Over-dispersion, entropy, and information criteria. Both the beta-binomial and gamma-Poisson models are maximum entropy for the same constraints as the regular binomial and Poisson. They just try to account for unobserved heterogeneity in probabilities

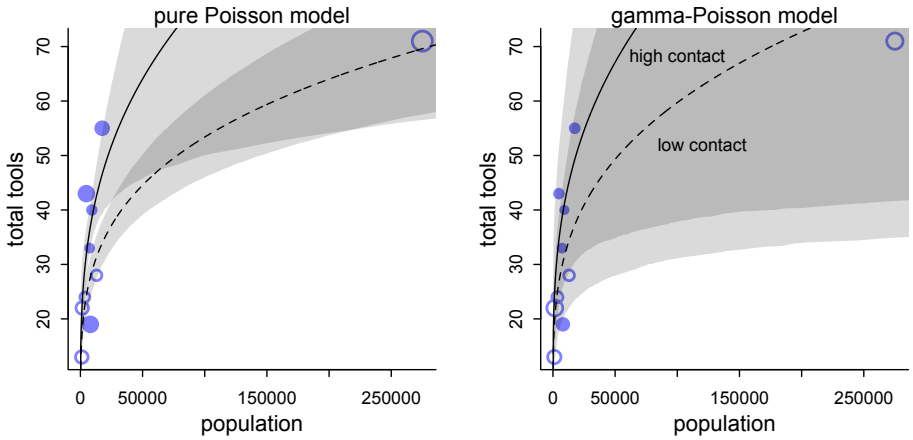

FIGURE 12.2. The Poisson model of Oceanic tools (left) is highly influenced by Hawaii. The equivalent gamma-Poisson model (right) is much less influenced by Hawaii, because the model expects more variation around the mean. And you can see the increased variation in the size of the shaded regions.

and rates. So while they can be a lot harder to fit to data, they can be usefully conceptualized much like ordinary binomial and Poisson GLMs. So in terms of model comparison using information criteria, a beta-binomial model is a binomial model, and a gamma-Poisson (negative-binomial) is a Poisson model.

You should not use WAIC and PSIS with these models, however, unless you are very sure of what you are doing. The reason is that while ordinary binomial and Poisson models can be aggregated and disaggregated across rows in the data, without changing any causal assumptions, the same is not true of beta-binomial and gamma-Poisson models. The reason is that a beta-binomial or gamma-Poisson likelihood applies an unobserved parameter to each row in the data. When we then go to calculate log-likelihoods, how the data are structured will determine how the beta-distributed or gamma-distributed variation enters the model.

For example, a beta-binomial model like the one examined earlier in this chapter has counts on each row. The rows were combinations of departments and gender in that case, and all of the applications for each department/gender combination were assumed to have the same unknown baseline probability of acceptance. What we'd like to do is treat each application as an observation, calculating WAIC over applications, so we get an estimate of accuracy for a new application to a known department/gender. We could disaggregate the data so each row is a single application. But if we do that, then we lose the fact that the beta-binomial model implies the same latent probability for all of the applicants from the same row in the data. This is a huge bother.

What to do? Once you see how to incorporate over-dispersion with multilevel models, in the next chapter, this obstacle will be reduced. Why? Because a multilevel model can assign heterogeneity in probabilities or rates at any level of aggregation.

Overthinking: Continuous mixtures. A distribution like the beta-binomial is called a *continuous*

mixture, because every binomial count is assumed to have its own independent beta-distributed probability of success, and the beta distribution is continuous rather than discrete. So the parameters of the beta-binomial are just the number of draws in each case (the same as the "size" n of the ordinary binomial distribution) and the two parameters that describe the shape of the beta distribution. All of this implies that the probability of observing a number of successes y from a beta-binomial process is:

$$f(y|n,\bar{p},\theta) = \int_0^1 g(y|n,p)h(p|\bar{p},\theta)dp$$

where f is the beta-binomial density, g is the binomial distribution, and h is the beta density. The integral above, like most integrals in applied probability, just computes an average: the probability of y, averaged over all values of p. The p values are drawn from the beta distribution with mean $\bar{p}$ and scale θ. The probability of a success p is no longer a free parameter, as it is produced by the beta distribution. The gamma-Poisson density has a similar form, but averaging a Poisson probability over a gamma distribution of rates.

In the case of the beta-binomial, as well as the gamma-Poisson, it is possible to close the integral above. You can look up the closed-form expressions anytime you need the analytic forms. The R functions `dbetabinom` and `dgampois` provide computations from them.

12.2. Zero-inflated outcomes

Very often, the things we can measure are not emissions from any pure process. Instead, they are *mixtures* of multiple processes. Whenever there are different causes for the same observation, then a MIXTURE MODEL may be useful. A mixture model uses more than one simple probability distribution to model a mixture of causes. In effect, these models use more than one likelihood for the same outcome variable.

Count variables are especially prone to needing a mixture treatment. The reason is that a count of zero can often arise more than one way. A "zero" means that nothing happened, and nothing can happen either because the rate of events is low or rather because the process that generates events failed to get started. If we are counting scrub jays in the woods, we might record a zero because there were no scrub jays in the woods or rather because we scared them all off before we starting looking. Either way, the data contains a zero.

So in this section you'll see how to construct simple zero-inflated models. You'll be able to use the same components from earlier models, but they'll be assembled in a different way. So even if you never need to use or interpret a zero-inflated model, seeing how they are constructed should expand your modeling imagination.

Rethinking: Breaking the law. In the sciences, there is sometimes a culture of anxiety surrounding statistical inference. It used to be that researchers couldn't easily construct and study their own custom models, because they had to rely upon statisticians to properly study the models first. This led to concerns about unconventional models, concerns about breaking the laws of statistics. But statistical computing is much more capable now. Now you can imagine your own generative process, simulate data from it, write the model, and verify that it recovers the true parameter values. You don't have to wait for a mathematician to legalize the model you need.

12.2.1. Example: Zero-inflated Poisson. Back in Chapter 11, I introduced Poisson GLMs by using the example of a monastery producing manuscripts. Each day, a large number of monks finish copying a small number of manuscripts. The process is essentially binomial,

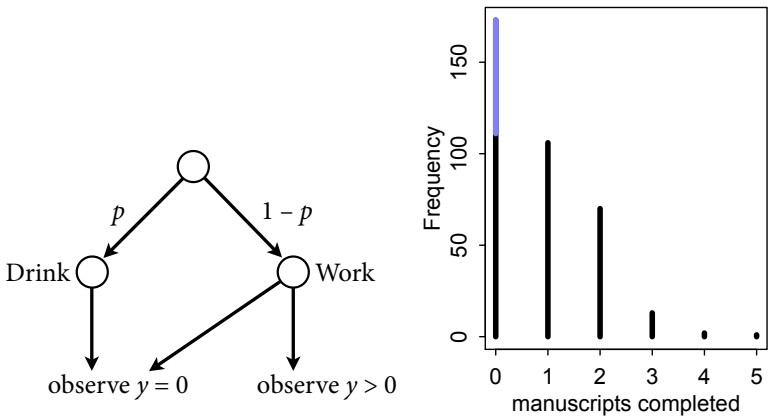

FIGURE 12.3. Left: Structure of the zero-inflated likelihood calculation. Beginning at the top, the monks drink p of the time or instead work $1 - p$ of the time. Drinking monks always produce an observation $y = 0$. Working monks may produce either $y = 0$ or $y > 0$. Right: Frequency distribution of zero-inflated observations. The blue line segment over zero shows the $y = 0$ observations that arose from drinking. In real data, we typically cannot see which zeros come from which process.

but with a large number of trials and very low probability, so the distribution tends towards Poisson.

Now imagine that the monks take breaks on some days. On those days, no manuscripts are completed. Instead, the wine cellar is opened and more earthly delights are practiced. As the monastery owner, you'd like to know how often the monks drink. The obstacle for inference is that there will be zeros on honest non-drinking days, as well, just by chance. So how can you estimate the number of days spent drinking?

Let's make a mixture to solve this problem.[178] We want to consider that any zero in the data can arise from two processes: (1) the monks spent the day drinking and (2) they worked that day but nevertheless failed to complete any manuscripts. Let p be the probability the monks spend the day drinking. Let λ be the mean number of manuscripts completed, when the monks work.

To get this model going, we need to define a likelihood function that mixes these two processes. To grasp how we can construct such a monster, think of the monks' drinking as resulting from a coin flip (FIGURE 12.3). The "coin" shows a cask of wine on one side and a quill on the other. The probability the wine cask shows is p, which could be any value from 0 to 1. Depending upon the outcome of the coin flip, the monks either begin drinking or rather begin copying. Drinking monks always produce zero completed manuscripts. Working monks produce a Poisson number of completed manuscripts with some average rate λ. So it is possible still to observe a zero, even when the monks work.

With these assumptions, the likelihood of observing a zero is:

$$\begin{aligned}\Pr(0|p,\lambda) &= \Pr(\text{drink}|p) + \Pr(\text{work}|p) \times \Pr(0|\lambda) \\ &= p + (1-p)\exp(-\lambda)\end{aligned}$$

Since the Poisson likelihood of y is $\Pr(y|\lambda) = \lambda^y \exp(-\lambda)/y!$, the likelihood of $y = 0$ is just $\exp(-\lambda)$. The above is just the mathematics for:

> *The probability of observing a zero is the probability that the monks didn't drink OR (+) the probability that the monks worked AND (×) failed to finish anything.*

And the likelihood of a non-zero value y is:

$$\Pr(y|y>0,p,\lambda) = \Pr(\text{drink}|p)(0) + \Pr(\text{work}|p)\Pr(y|\lambda) = (1-p)\frac{\lambda^y \exp(-\lambda)}{y!}$$

Since drinking monks never produce $y > 0$, the expression above is just the chance the monks both work, $1-p$, and finish y manuscripts.

Define ZIPoisson as the distribution above, with parameters p (probability of a zero) and λ (mean of Poisson) to describe its shape. Then a zero-inflated Poisson regression takes the form:

$$\begin{aligned}y_i &\sim \text{ZIPoisson}(p_i, \lambda_i) \\ \text{logit}(p_i) &= \alpha_p + \beta_p x_i \\ \log(\lambda_i) &= \alpha_\lambda + \beta_\lambda x_i\end{aligned}$$

Notice that there are two linear models and two link functions, one for each process in the ZIPoisson. The parameters of the linear models differ, because any predictor such as x may be associated differently with each part of the mixture. In fact, you don't even have to use the same predictors in both models—you can construct the two linear models however you wish, depending upon your hypothesis.

We have everything we need now, except for some data. So let's simulate the monks' drinking and working. Then you'll see the code used to recover the parameter values used in the simulation.

R code 12.7

```
# define parameters
prob_drink <- 0.2 # 20% of days
rate_work <- 1    # average 1 manuscript per day

# sample one year of production
N <- 365

# simulate days monks drink
set.seed(365)
drink <- rbinom( N , 1 , prob_drink )

# simulate manuscripts completed
y <- (1-drink)*rpois( N , rate_work )
```

The outcome variable we get to observe is `y`, which is just a list of counts of completed manuscripts, one count for each day of the year. Take a look at the outcome variable:

R code 12.8

```
simplehist( y , xlab="manuscripts completed" , lwd=4 )
zeros_drink <- sum(drink)
zeros_work <- sum(y==0 & drink==0)
zeros_total <- sum(y==0)
lines( c(0,0) , c(zeros_work,zeros_total) , lwd=4 , col=rangi2 )
```

This plot is shown on the right-hand side of FIGURE 12.3. The zeros produced by drinking are shown in blue. Those from work are shown in black. The total number of zeros is inflated, relative to a typical Poisson distribution.

And to fit the model, the `rethinking` package provides the zero-inflated Poisson likelihood as `dzipois`. For more detail on how it relates to the mathematics above, see the Overthinking box at the end of this section. Using `dzipois` is straightforward. I'm also going to nudge the prior for the probability of drinking so that there is more mass below 0.5 than above it—the monks probably do not drink more often than not.

R code 12.9

```
m12.3 <- ulam(
    alist(
        y ~ dzipois( p , lambda ),
        logit(p) <- ap,
        log(lambda) <- al,
        ap ~ dnorm( -1.5 , 1 ),
        al ~ dnorm( 1 , 0.5 )
    ) , data=list(y=y) , chains=4 )
precis( m12.3 )
```

```
    mean   sd  5.5% 94.5% n_eff Rhat
ap -1.28 0.35 -1.89 -0.79   657    1
al  0.01 0.09 -0.14  0.16   759    1
```

On the natural scale, those posterior means are:

R code 12.10

```
post <- extract.samples( m12.3 )
mean( inv_logit( post$ap ) ) # probability drink
mean( exp( post$al ) )       # rate finish manuscripts, when not drinking
```

```
[1] 0.2241255
[1] 1.017643
```

Notice that we can get an accurate estimate of the proportion of days the monks drink, even though we can't say for any particular day whether or not they drank.

This example is the simplest possible. In real problems, you might have predictor variables that are associated with one or both processes inside the zero-inflated Poisson mixture. In that case, you add those variables and their parameters to either or both linear models.

Overthinking: Zero-inflated Poisson calculations in Stan. The function `dzipois` is implemented in a way that guards against some kinds of numerical error. So its code looks confusing—just type "`dzipois`" at the R prompt and see. But really all it's doing is implementing the likelihood formula defined in the section above. Let's focus on how this is implemented in Stan. When you tell `ulam` to use `dzipois`, it understands it like this:

R code 12.11

```
m12.3_alt <- ulam(
    alist(
        y|y>0 ~ custom( log1m(p) + poisson_lpmf(y|lambda) ),
        y|y==0 ~ custom( log_mix( p , 0 , poisson_lpmf(0|lambda) ) ),
        logit(p) <- ap,
        log(lambda) <- al,
        ap ~ dnorm(-1.5,1),
        al ~ dnorm(1,0.5)
    ) , data=list(y=as.integer(y)) , chains=4 )
```

That is the same model, but with explicit mixtures and some raw Stan code inside the `custom` lines. If you look at `stancode(m12.3_alt)`, you'll see the corresponding lines:

```
if ( y[i] > 0 ) target += log1m(p) + poisson_lpmf(y[i] | lambda);
if ( y[i] == 0 ) target += log_mix(p, 0, poisson_lpmf(0 | lambda));
```

That `target` thing is a chain of terms for calculating the log-posterior. When we use it with `+=`, we add another term to the stack. Stan will later use this stack to figure out the gradient, through aggressive and systematic use of the chain rule from calculus. Then there are some important tricks for doing this calculation. The `log1m` function computes the log of one-minus a value. We need $\log(1-p)$, but if p is very close to 1, then this can round catastrophically to zero and then the log will be negative infinity. Using `log1m` makes this much less likely. The function `log_mix` mixes together two log-probabilities, which is what we need for the probability of a zero. But it also uses clever techniques to avoid rounding error. It's equivalent in this case to:

```
if ( y[i] == 0 ) target += log( p + (1-p)*exp(-lambda) );
```

but more stable under extreme values of `p`. In this case, it makes no difference—the less fancy direct approach works fine. But it's good to know the better approach. You'll see it the code and more complex models won't work right otherwise. Finally, note that I coerced `y` to integer in the data list. When you use `ulam`'s built-in distributions, it will try to coerce variables into the correct Stan type. But if you build your own, you need to do this yourself.

12.3. Ordered categorical outcomes

It is very common in the social sciences, and occasional in the natural sciences, to have an outcome variable that is discrete, like a count, but in which the values merely indicate different ordered levels along some dimension. For example, if I were to ask you how much you like to eat fish, on a scale from 1 to 7, you might say 5. If I were to ask 100 people the same question, I'd end up with 100 values between 1 and 7. In modeling each outcome value, I'd have to keep in mind that these values are *ordered*, because 7 is greater than 6, which is greater than 5, and so on. The result is a set of **ORDERED CATEGORIES**. Unlike a count, the differences in value are not necessarily equal. It might be much harder to move someone's preference for fish from 1 to 2 than it is to move it from 5 to 6. Just treating ordered categories as continuous measures is not a good idea.[179]

Luckily, there is a standard and accessible solution. In principle, an ordered categorical variable is just a multinomial prediction problem (page 366). But the constraint that the categories be ordered demands a special treatment. What we'd like is for any associated predictor variable, as it increases, to move predictions progressively through the categories in sequence. So for example if preference for ice cream is positively associated with years of age, then the model should sequentially move predictions upwards as age increases: 3 to 4, 4 to 5, 5 to 6, etc. This presents a challenge: how to ensure that the linear model maps onto the outcomes in the right order.