16 Generalized Linear Madness

When I asked my high school physics teacher about statistics, she told me a joke. Here's how I remember it. A physicist, an engineer, and a statistician go bow hunting together. After many hours, they spot a deer in the distance. The physicist does a quick ballistic calculation, ignoring air resistance. The arrow flies true but falls a few meters short of the target. The deer doesn't notice. The engineer smirks, introduces a fudge factor for air resistance, and shoots. The second arrow lands instead a few meters long. The deer still doesn't notice. The statistician takes the average and yells, "We got it!"

The sciences construct theories of natural processes. Eventually these theories are expressed formally, as mathematical models. Such models are specialized, make precise predictions, and can fail in equally precise ways. Being wrong in precise ways is useful, because the failures borrow meaning from the cause and effect relationships built into the models. This is true of the physicist and the engineer in the joke. They were wrong in very precise ways that give us hints about which causes were at fault.

Applied statistics has to apply to all the sciences, and so it is often much vaguer about models. Instead it focuses on average performance, regardless of the model. The generalized linear models in the preceding chapters are not credible scientific models of most natural processes. They are powerful, geocentric (Chapter 4) descriptions of associations. In combination with a logic of causal inference, for example DAGs and *do*-calculus, generalized linear models can nevertheless be unreasonably powerful.

But there are problems with this GLMs-plus-DAGs approach. Not everything can be modeled as a GLM—a linear combination of variables mapped onto a non-linear outcome. But if it is the only approach you know, then you have to use it. Other times the theory of interest can be expressed as a GLM, but the theory implies that some of the parameters are fixed at special values. We might never notice, if we start with GLMs instead of real models. And when a GLM fails, it's not easy to learn from the failure. Debugging epicycles is a game no one can win. If we could replace the heuristic DAG with an actual structural causal model, we might solve all these problems at once.

In this chapter, I will go beyond GENERALIZED LINEAR MADNESS. I'll work through examples in which the scientific context provides a causal model that will breathe life into the statistical model. I've chosen examples which are individually distinct and highlight different challenges in developing and translating causal models into *bespoke* (see the Rethinking box below) statistical models. You won't require any specialized scientific expertise to grasp these examples. And the basic strategy is the same as it has been from the start: Define a generative model of a phenomenon and then use that model to design strategies for causal inference and statistical estimation.

Unlike the other chapters in this book, there is some mathematics in this chapter, and it really cannot be avoided. But all you need is some algebra. We won't so much do math as express ideas with math. We will also work directly with Stan model code, since `ulam()` is not flexible enough for some of the examples. If you aren't interested in the code, you can ignore it. But as usual, seeing the implementation often helps to clarify the concepts.

Rethinking: Bespoken for. Mass production has some advantages, but it also makes our clothes fit badly. Garments bought off-the-shelf are not manufactured with you in mind. They are not *bespoke* products, designed for any particular person with a particular body. Unless you are lucky to have a perfectly average body shape, you will need a tailor to get better.

Statistical analyses are similar. Generalized linear models are off-the-shelf products, mass produced for a consumer market of impatient researchers with diverse goals. Science asked statisticians for tools that could be used anywhere. And so they delivered. But the clothes don't always fit.

One problem with off-the-shelf models is that they interrupt expertise. A typical researcher knows a lot about their subject. Evidence of this is the detailed objections a scientist makes when someone from another specialty tries to build a theoretical model for their subject. But then when those scientists turn to analyze their own data, they use tools that forbid the use of that knowledge. There is no way in a standard GLM to incorporate it. Even worse, if the only models researchers are ever taught are GLMs (or GLMMs), these models may crowd out the formation of informed, bespoke scientific models.

GLMs are unreasonably powerful. But we should remember that they are usually only geocentric devices. Better bespoke models are eventually necessary, both for better fit and better inference.

16.1. Geometric people

Back in Chapter 4, you met linear regression in the context of building a predictive model of height using weight. You even saw how to measure non-linear associations between the two variables. But nothing in that example was scientifically satisfying. The height-weight model was just a statistical device. It contains no biological information and tells us nothing about how the association between height and weight arises. Consider for example that weight obviously does not *cause* height, at least not in humans. If anything, the causal relationship is the reverse.

So now let's try to do better. Why? Because when the model is scientifically inspired, rather than just statistical required, disagreements between model and data are informative of real causal relationships.

Suppose for example that a person is shaped like a cylinder. Of course a person isn't exactly shaped like a cylinder. There are arms and a head. But let's see how far this cylinder model gets us. The weight of the cylinder is a consequence of the volume of the cylinder. And the volume of the cylinder is a consequence of growth in the height and width of the cylinder. So if we can relate the height to the volume, then we'd have a model to predict weight from height.

16.1.1. The scientific model.

Let's do it. Sometime a long time ago you learned, and sensibly forgot, that the formula for the volume of a person-cylinder is:

$$V = \pi r^2 h$$

where r is the person's radius and h is its height. See FIGURE 16.1.[219] We don't know each individual's radius, but let's assume that each individual's radius is some constant proportion

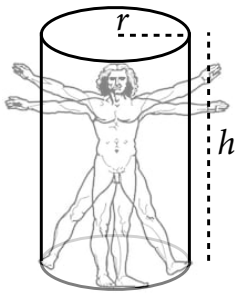


FIGURE 16.1. The "Vitruvian Can" model of human weight as a function of height. If Vitruvian Man were a cylinder, we could estimate his weight by calculating his volume V as a function of his height h and radius r.

p of height. This means $r = ph$. Substituting this into the formula:

$$V = \pi(ph)^2 h = \pi p^2 h^3$$

Finally, weight is some proportion of volume—how many kilograms are there per cubic centimeter? So we need a parameter k that expresses this translation between volume and weight.

$$W = kV = k\pi p^2 h^3$$

And this is our formula for expected weight, given an individual's height h. This is not obviously an ordinary generalized linear model. But that's okay. It makes predictions, and we can fit it to data.

Rethinking: Spherical cows. Useful mathematical modelling typically involves ridiculous assumptions. For example, the assumption above that people are shaped like cylinders. This type of assumption can be called a SPHERICAL COW, after the book *Consider a Spherical Cow: A Course in Environmental Problem Solving*.[220] Strategic, simplifying assumptions are features of all useful models. By first understanding the simplified model, it is easier to later add in relevant detail, where the flaws in the simpler model help us decide which details are relevant. Non-mathematical models are also simplifications, but usually the simplifications are not explicit. This makes it harder to identify their flaws.[221]

16.1.2. The statistical model. We can use the cylinder formula in a statistical model. To do so however, we need to make some more choices. Here's the model outline. I'll explain each piece afterwards.

$$\begin{aligned} W_i &\sim \text{Log-Normal}(\mu_i, \sigma) && \text{[Distribution for weight]} \\ \exp(\mu_i) &= k\pi p^2 h_i^3 && \text{[expected median of weight]} \\ k &\sim \textit{some prior} && \text{[prior relation between weight and volume]} \\ p &\sim \textit{some prior} && \text{[prior proportionality of radius to height]} \\ \sigma &\sim \text{Exponential}(1) && \text{[our old friend, sigma]} \end{aligned}$$

From the top, the first thing to decide is the distribution for the observed outcome variable, weight W_i. This variable is positive—weight can't be negative—and continuous. So I've chosen a Log-Normal distribution. The Log-Normal distribution is parameterized by the mean of the logarithm, which is called μ_i. The median of the Log-Normal is $\exp(\mu_i)$. In the model

above, I've assigned this median to be the cylinder function. Finally, we need priors for the three parameters k, p, and σ.

One of the major advantages of having a scientifically inspired model is that the parameters have meanings. These meanings constitute prior information that we can use to chose informative distributions. This is especially useful in these contexts, because often there are more scientifically-required parameters than can be directly identified by the data. We can nevertheless do useful estimation, given some scientific constraints on the parameters. That is the case in this example.

The first thing to notice about the parameters k and p is that they are multiplied in the model and the data have no way to estimate anything except their product. The technical way this problem could be described is that k and p, given this model and these data, are not **IDENTIFIABLE**. We could just replace the product kp^2 with a new parameter θ and estimate that instead. Like this:

$$\exp(\mu_i) = \pi\theta h_i^3$$

We'll get the same predictions. What we won't get is an easy way to assign a prior to θ. So even if we are going to use $\theta = kp^2$ trick, we'll need to think still about k and p.

Let's think about the parameter p. It is the ratio of the radius to the height, $p = r/h$. So it must be greater than zero. It must also be less than one, because few people are wider than they are tall. It is almost certainly less than one-half, because a person as wide as they are tall would have $2r = h$, making $p = (h/2)/h = 0.5$. So p is probably much less than 0.5. Putting all of this together, what we want is a distribution bounded between zero and one with most of the prior mass below 0.5. A beta distribution will do:

$$p \sim \text{Beta}(2, 18)$$

This prior will have mean $2/(2 + 18) = 0.1$. We really need to do some prior predictive simulations to do better (see the practice problems at the end of this chapter). But that takes care of p for the moment.

The parameter k is the proportion of volume that is weight. It really just translates measurement scales, because changing the units of volume or weight will change its value. For example, if height is measured in centimeters and weight is measured in kilograms, then volume has units cm^3, and so k must have units kg/cm^3. The definition of k, in that case, is just how many kilograms there are per cubic centimeter. So to scale the prior right, we need to have some information about how heavy a cubic centimeter of person is. We could look that up, or maybe use our own bodies to get a prior.

Rethinking: Priors are never arbitrary. It's commonplace to hear the fearful claim that Bayes is untrustworthy because priors are arbitrary. It is true that people sometimes treat priors that way. But priors are only arbitrary when scientists ignore domain knowledge. Even when we stick with GLMs, prior predictive simulations force us to engage with background knowledge to produce useful, non-arbitrary priors. When we have a more scientifically grounded model, the parameters have even more meaning. The p and k parameters in the cylinder example have scientific meanings that let us assign priors that could even be measured physically. Using flat priors in this example, out of some metaphysical commitment to ignorance, would be a mistake.

But suppose you couldn't look it up. What then? A very useful trick is to instead get rid of the measurement scales altogether—measurement scales are arbitrary human inventions—and then use the known biological constraints to locate the prior. How do we get rid of measurement scale? We can divide the observed variables by some reference values. This will divide out the units. For example, suppose that we divide both height and weight by their mean values.

R code 16.1

```
library(rethinking)
data(Howell1)
d <- Howell1

# scale observed variables
d$w <- d$weight / mean(d$weight)
d$h <- d$height / mean(d$height)
```

The new variables `w` and `h` have means of 1. There is nothing special about using the means here. We just need some reference value to divide out the units. Now consider what a plausible value of k might be, under this scaling. Suppose we have an individual of average height and weight. In that case $w_i = 1$ and $h_i = 1$. Plugging these into the formula:

$$1 = k\pi p^2 1^3$$

Assuming $p < 0.5$, then k must be greater than 1. I suggest we constrain k to be positive (it has to be) and give it a prior mean around 2.

$$k \sim \text{Exponential}(0.5)$$

We could certainly do better than this, with some prior predictive simulation. But this will get us started.

Now let's pull all the threads together into a tapestry of code.

R code 16.2

```
m16.1 <- ulam(
    alist(
        w ~ dlnorm( mu , sigma ),
        exp(mu) <- 3.141593 * k * p^2 * h^3,
        p ~ beta( 2 , 18 ),
        k ~ exponential( 0.5 ),
        sigma ~ exponential( 1 )
    ), data=d , chains=4 , cores=4 )
```

Take a look at the `precis()` output. Can you make sense of the posterior distributions of p and k? How were the priors updated?

While you think of answers to those questions, Let's inspect what the posterior does with the lack of identifiability of k and p. The `pairs(m16.1)` plot is the easiest way to appreciate it. I show this plot in FIGURE 16.2, on the left. There is a narrow curved ridge in the posterior where combinations of k and p produce the same product kp^2. This results in a strong negative correlation between the two parameters—if one gets bigger, the other has to get smaller to maintain the same product. Because we used informative priors, we were able to fit this model anyway. But there is still no independent information about these parameters in the data itself. At least not with this model. There's no reason in principle that k and p aren't also functions of height (or age). For example, muscle and far have very different densities.

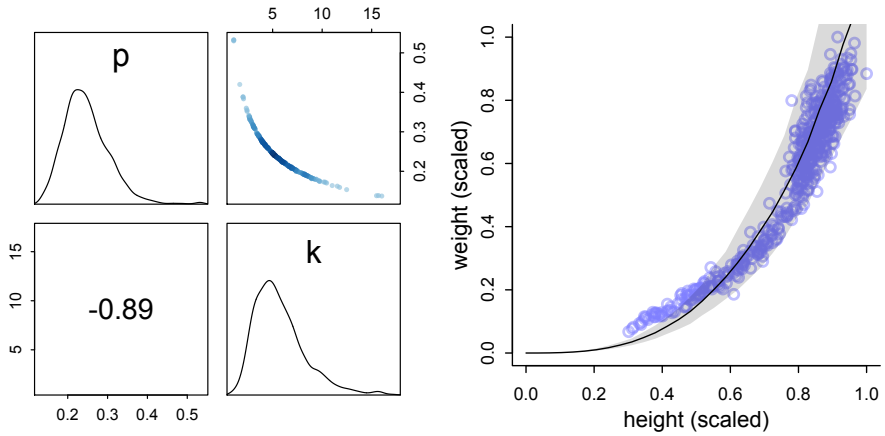


FIGURE 16.2. Left: Posterior distribution of k and p. Because only the product kp^2 appears in the model definition, the data alone cannot identify k and p, but only the product. The prior distributions make estimation possible. Right: The cylinder model fit to the !Kung data. Note the poor fit at short heights.

So k isn't necessarily a constant, because relative muscle mass isn't a constant. Similarly, the ratio of body width to height isn't constant over development. So p may change as well.

The idea that p may change can help us understand the posterior predictions. Let's plot the posterior predictive distribution across the observed range of height.

R code 16.3

```
h_seq <- seq( from=0 , to=max(d$h) , length.out=30 )
w_sim <- sim( m16.1 , data=list(h=h_seq) )
mu_mean <- apply( w_sim , 2 , mean )
w_CI <- apply( w_sim , 2 , PI )
plot( d$h , d$w , xlim=c(0,max(d$h)) , ylim=c(0,max(d$w)) , col=rangi2 ,
    lwd=2 , xlab="height (scaled)" , ylab="weight (scaled)" )
lines( h_seq , mu_mean )
shade( w_CI , h_seq )
```

The result is displayed in the right panel of FIGURE 16.2. First, note that the model gets the general scaling relationship right. The exponent on height is fixed by theory at 3. We didn't estimate it. But it does a great job. Second, note the poor fit for the smallest heights in the sample. The is possibly a symptom of p being different for children, as well as possibly k. The important lesson is that misfit for a scientific model gives us useful hints. If this were just a linear regression, the parameters wouldn't have biological meanings and we would fix it by spinning up some epicycles.

16.1.3. GLM in disguise. Before moving on to the next example, consider what happens to this model when we relate the logarithm of weight to height. In that case, the expectation is:

$$\log w_i = \mu_i = \log(k\pi p^2 h_i^3)$$

Now since multiplication becomes addition on the log scale, we can rewrite this as:

$$\log w_i = \log(k) + \log(\pi) + 2\log(p) + 3\log(h_i)$$

On the log scale, this is a linear regression. The first three terms above comprise the intercept. Then the term $3\log(h_i)$ is a predictor variable with a fixed coefficient of 3. Theory gave us the value of that coefficient. We didn't need to estimate it. But it still has the form of an ordinary linear regression term.

I point this out to highlight one of the reasons that generalized linear models are so powerful. Lots of natural relationships are GLM relationships, on a speicfic scale of measurement. At the same time, the GLM approach wants to simply estimate parameters which may be informed by a proper theory, as in this case.

16.2. Hidden minds and observed behavior

One of the most basic problem in scientific inference is the so-called **INVERSE PROBLEM**: How to figure out causes from observations. It is a problem, because many different causes can produce the same evidence. So while it can be easy to go forward from a known cause to predicted observations, it can very hard to go backwards from observation to cause.

Every branch of science has its own inverse problems. In this section, we'll consider a simple example from developmental psychology. Children may possess many different cognitive strategies for making decisions. Given some observations of their behavior, which strategy was the cause? Let's consider specifically an experiment in which 629 children aged 4 to 14 saw four other children choose among three different colored boxes (FIGURE 16.3). Each child then made their own choice. In each trial, three demonstrators choose the same color. The fourth demonstrator chose a different color. So in each trial, one of the colors was the majority choice, another was the minority choice, and the final color was unchosen. How do we figure out from this experiment whether children are influenced by the majority?

Let's load the data[222] and take a closer look.

R code 16.4

```
library(rethinking)
data(Boxes)
precis(Boxes)
```

FIGURE 16.3. The apparatus used in the experiment. The "choice box" has three tubes, each with a different color. When a ball is dropped into a tube, a toy comes out of the box. Four children demonstrated. Then the choice of a fifth child was recorded. How did the choices of the first four influence the fifth child's choice?

```
'data.frame': 629 obs. of 5 variables:
               mean   sd 5.5% 94.5%      histogram
y              2.12 0.73    1     3
male           0.51 0.50    0     1
age            8.03 2.50    5    13
majority_first 0.48 0.50    0     1
culture        3.75 1.96    1     8
```

The outcome `y` here takes the values 1, 2, and 3. It indicates which of the three options were chosen, where 1 indicates the unchosen color, 2 indicates the majority demonstrated color, and 3 indicates the minority demonstrated color. The other variable that we'll use in this example is `majority_first`, which indicates whether the majority color was demonstrated before the minority color. This is counter balanced across trials. The other variables are also interesting. But let's set them aside for the moment.

We're interested in using the outcome `y` to infer the strategies the children used to choose a color. The distribution of the outcome contains 45% majority color choices:

R code 16.5

```
table( Boxes$y ) / length( Boxes$y )
```

```
        1         2         3
0.2114467 0.4562798 0.3322734
```

Does this mean that 45% of the children used the strategy of following the majority? No. The core inferential problem is that there are three choices and many possible strategies. And different strategies can produce the same choice in the same trial. For example, a child could just choose at random. This will result one-third of the time in the same prediction as a child who follows the majority. A GLM of these choices would infer frequencies behavior. But we want to make inferences about strategy. How can we do this?

16.2.1. The scientific model. The key, as always, is to think generatively. Consider for example a group of children in which half of them choose at random and the other half follow the majority. If we simulate choices for these children, we can figure out how often we might see the "2" choice, the one that indicates the majority color.

R code 16.6

```
set.seed(7)
N <- 30 # number of children

# half are random
# sample from 1,2,3 at random for each
y1 <- sample( 1:3 , size=N/2 , replace=TRUE )

# half follow majority
y2 <- rep( 2 , N/2 )

# combine and shuffle y1 and y2
y <- sample( c(y1,y2) )

# count the 2s
sum(y==2)/N
```

```
[1] 0.6333333
```

About two-thirds of the choices are for the majority color, but only half the children are actually following the majority. The above is only one simulation, but it demonstrates the problem. When different hidden strategies can produce the same behavior, inference about strategy is more complicated than just counting behavior.

We'll consider 5 different strategies children might use.

(1) Follow the Majority: Copy the majority demonstrated color.
(2) Follow the Minority: Copy the minority demonstrated color.
(3) Maverick: Choose the color that no demonstrator chose.
(4) Random: Choose a color at random, ignoring the demonstrators.
(5) Follow First: Copy the color that was demonstrated first. This was either the majority color (when `majority_first` equals 1) or the minority color (when 0).

Each strategy entails a vector of three probabilities, one for each choice. For example, Random is $[1/3, 1/3, 1/3]$. The complicated one is Follow First, which depends upon the order of presentation.

An obvious question is: Why these strategies? The equally obvious answer is: Because they seem *a priori* plausible. If there are some that you think are not plausible, or yet more strategies that you feel should be added, the same generative framework can accomodate them.

16.2.2. The statistical model. Now we need a statistical model that reflects the generative model above. Remember, statistical models run in reverse of generative models. In the generative model, we assume strategies and simulate observed behavior. In the statistical model, we instead assume observed behavior (the data) and simulate strategies (parameters).

In this example, we can't directly measure each child's strategy. It is an unobserved variable. But each strategy has a specific probability of producing each choice. We can use that fact to compute the probability of each choice, given parameters which specify the probability of each strategy. Then we let Bayes loose and get the posterior distribution of each strategy back. Before we can let Bayes loose, we'll need to enumerate the parameters, assign priors to each, and also figure out some technical issues for coding. I'll move through these tasks slowly.

The unobserved variables are the probabilities that a child uses each of the five strategies. This means five values, but since these must sum to one, we need only four parameters. There is a variable type called a **SIMPLEX** that handles this for us. A simplex is a vector of values that must sum to some constant, usually one. Stan allows us to declare a vector of parameters as a simplex, and then Stan handles the bookkeeping of the constant sum for us. We can give this simplex a Dirichlet prior, which is a prior for probability distributions. We used both Dirichlet and a simplex already back in Chapter 12 to construct ordered categorical predictors (page 405). We'll use a weak uniform prior on the simplex of strategy probabilities, which we'll label p:

$$p \sim \text{Dirichlet}([4, 4, 4, 4, 4])$$

As you saw back in Chapter 12, this prior doesn't mean that we expect the strategies to be equally probable. Instead it means that we expect that any one of them could be more or less probable than any other. If you make those 4s larger, the prior starts to say that we expect them to be actually equal.

Now how to express the probability of the data, the likelihood? For each observed choice y_i, each strategy s implies a probability of seeing y_i. Call this $\Pr(y_i|s)$, the probability of the

data, conditional on assuming a specific strategy s. For example assuming $s = 1$, the majority strategy, then $\Pr(y_i = 2|s = 1) = 1$. This is just the mathy way of saying that a child using the majority strategy always follows the majority color choice.

We don't know s though. We can't observe it directly. However we do have a probability for each s in the model. These are the elements of the simplex p. So to get the unconditional probability of the data $\Pr(y_i)$ we just need to use p to average over the unknown strategy s:

$$\Pr(y_i) = \sum_{s=1}^{5} p_s \Pr(y_i|s)$$

Read this as *the probability of y_i is the weighted average of the probabilities of y_i conditional on each strategy s*. This expression is a mixture, as in earlier chapters. Sometimes you'll read that this *marginalizes* out the unknown strategy. This just means averaging over the strategies, using some probability of each to get the weight of each in the average. Above, the values in p provide these weights.

Okay, so we have our statistical model now. Let's write it in a more conventional form:

$$\begin{aligned} y_i &\sim \text{Categorical}(\theta) \\ \theta_j &= \sum_{s=1}^{5} p_s \Pr(j|s) \qquad \text{for } j = 1\ldots3 \\ p &\sim \text{Dirichlet}([4,4,4,4,4]) \end{aligned}$$

The vector θ holds the average probability of each behavior, conditional on p. As a generative model, the above implies that all children are identical—each child on each trial has some probability p_s of using strategy s. Of course there are individual differences among the children. But since we don't have any repeat observations of each child in these data, we can't do much better than the above. But if we did have repeat observations, we'd assign a unique simplex p to each child, power up the partial pooling, and enjoy the fireworks.

16.2.3. Coding the statistical model. Coding this model means explicitly coding the logic of each strategy, those $\Pr(j|s)$ terms above. We will write this model directly in Stan, because it will actually make it both easier to code and easier to extend. There have been some optional Stan models in previous chapters. But now it's not optional. I've included the model code in the `rethinking` package. You can load and display it with:

R code 16.7

```
data(Boxes_model)
cat(Boxes_model)
```

I'll put the explanation of the Stan code in the Overthinking box further down, so you can focus on the coding details later.

To run the sampler, all that remains is to prepare the data list and then invoke `stan()`. The data list needs only the sample size `N`, the vector of choices `y`, and the vector of presentation order `majority_first`.

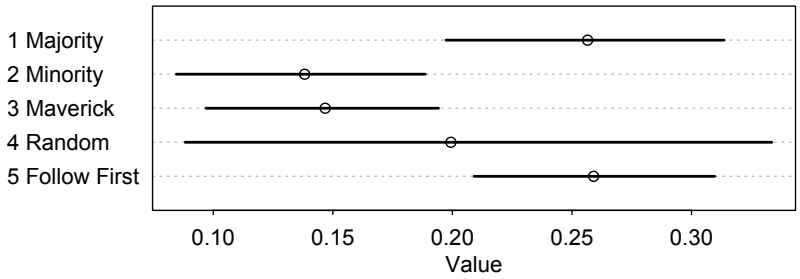
R code 16.8

```
# prep data
dat_list <- list(
    N = nrow(Boxes),
    y = Boxes$y,
```

```
    majority_first = Boxes$majority_first )

# run the sampler
m16.2 <- stan( model_code=Boxes_model , data=dat_list , chains=3 , cores=3 )

# show marginal posterior for p
p_labels <- c("1 Majority","2 Minority","3 Maverick","4 Random","5 Follow First")
plot( precis(m16.2,2) , labels=p_labels )
```



Recall that 45% of the sample chose the majority color. But the posterior distribution is consistent with somewhere between 20% and 30% of children following the majority copying strategy. Conditional on this model, a similar proportion just copied the first color that was demonstrated. This is what hidden state models can do for us—prevent us from confusing behavior with strategy.

This model can be extended to allow the probabilities of each strategy to vary by age, gender, or anything else. In principle, this is easy—you just make p_s conditional on the predictor variables. In practice, there are coding decisions to make. I say more about this in the Overthinking box below.

Overthinking: Stan code for the Boxes model. A Stan model needs three "blocks" of code. I'll explain each in order. The first block is the DATA BLOCK. This block just names the observed variables and declares their types. For this model, it looks like this:

```
data{
    int N;
    int y[N];
    int majority_first[N];
}
```

The integer `N` is just a count of observed cases. It's the number of rows in `data(Boxes)`. Then the outcome `y` and predictor `majority_first` are declared as integer vectors of length `N`. You could hard-code the length as the number 629. But then you have to change the model code every time the number of cases changes. The second block a Stan model needs is the PARAMETERS BLOCK. This is like the data block, but for unobserved variables. These are the variables that we get posterior samples for. In this model, it contains only the simplex `p`:

```
parameters{
    simplex[5] p;
}
```

The third block is the heart, the MODEL BLOCK. This block calculates the log-probability of the variables, both observed (data) and unobserved (parameters). This is the numerator in Bayes' theorem, and Stan uses it to run the Hamiltonian simulation (see Chapter 9). I'll take this block in pieces. At

the top, we declare a vector to hold probability calculations for each strategy. We'll reuse this vector on each row of the data, to compute different probabilities.

```
model{
    vector[5] phi;
```

Next we assign the prior.

```
    // prior
    p ~ dirichlet( rep_vector(4,5) );
```

Now the heart of the matter. We loop over all rows. For each row `i`, we compute the log-probability of the observed `y[i]`. Each strategy has its own `if...then` to assign the probability of the data, conditional on that strategy. This gives us:

```
    // probability of data
    for ( i in 1:N ) {
        if ( y[i]==2 ) phi[1]=1; else phi[1]=0; // majority
        if ( y[i]==3 ) phi[2]=1; else phi[2]=0; // minority
        if ( y[i]==1 ) phi[3]=1; else phi[3]=0; // maverick
        phi[4]=1.0/3.0;                         // random
        if ( majority_first[i]==1 )             // follow first
            if ( y[i]==2 ) phi[5]=1; else phi[5]=0;
        else
            if ( y[i]==3 ) phi[5]=1; else phi[5]=0;
```

Now we need to include the p parameters. We do this by adding each $\log(p_s)$ to the log-probabilities computed above. Then we add the average probability to the `target`, which is just Stan's name for the total log-probability.

```
        // compute log( p_s * Pr(y_i|s) )
        for ( s in 1:5 ) phi[s] = log(p[s]) + log(phi[s]);
        // compute average log-probability of y_i
        target += log_sum_exp( phi );
```

That `log_sum_exp` function computes the marginal log-probability of the data, $\log \Pr(y_i)$, as defined in the main text. `log_sum_exp` takes the `phi` vector, which contains the individual log-probabilities for each strategy, and returns the logarithm of their sum on the probability scale. It's used a lot in Stan models like this, models with discrete parameters.

To modify the model to include predictor variables, there are many options. So falling back again on some real theory will help to focus the effort. The simplest sort of modification is to allow the `p` simplex to vary by some discrete category, like gender. In that case, we add the variable `gender` to the data block and add a dimension to `p` in the parameters block, like this:

```
    simplex[5] p[2];
```

And then in the model block, just index `p` by both strategy and gender with `p[gender[i],s]`:

```
        for ( s in 1:5 ) phi[s] = log(p[gender[i],s]) + log(phi[s]);
```

This model is in the `rethinking` package as `data(Boxes_model_gender)`. A continuous covariate like `age` presents many more choices. Gaussian processes, splines, polynomials can all manage the job. Each must be coded a different way. The Stan model `data(Boxes_model_age)` shows a simple linear age trend example, in which each `p` is assigned a linear model on the logit scale, and these are transformed with multi-inverse-logit to the simplex scale. This is entirely geocentric. If you have a stronger theory, it helps.

16.2.4. State space models. The Boxes model above resembles a broader class of model known as a **STATE SPACE MODEL**. These models posit multiple hidden states that produce observations. Typically the states are dynamic, changing over time. When the states are discrete categories, the model may be called a **HIDDEN MARKOV MODEL** (HMM). Many time series models are state space models, since the true state of the time series is not observed, only the noisy measures. There is an example later in this chapter.

16.3. Ordinary differential nut cracking

The *Panda* nut has nothing to do with bears. It is a big, hard nut produced by the evergreen tree *Panda oleosa*. People have been eating delicious *Panda* nuts for millennia, cracking them open with stone and steel tools. Other animals have a harder time getting into these nuts. But the chimpanzees of Ivory Coast manage the same way people do, by using tools. The chimpanzees use stone and wooden hammers to open *Panda* nuts, and they do so with high efficiency.

In this section, we're going to model the development of nut opening skill among these chimpanzees. Let's load the data and outline the project:

R code 16.9

```
library(rethinking)
data(Panda_nuts)
```

These data are records of individual bouts of nut opening.[223] Each row is an individual-bout pair. The variables of immediate interest are the outcome `nuts_opened`, the duration pf the bout in `seconds`, and the individual's `age`. The research question is how nut opening skill develops and which factors contribute to it. One reason to care about this question is that tool use in primates is very rare. Yet humans cannot live without tools. How did we end up this way? Understanding the evolution of human technology benefits from species comparisons that tease apart the relative contributions of cognition, dexterity, social learning, and strength. We're not going to achieve all that in this section. But we will get started. And we won't use a GLM.

16.3.1. Scientific model. We need a generative model of nut opening rate as it varies by age. Let's consider the dumbest model, which is nevertheless smarter than a GLM. Suppose the only factor that matters is the individual's strength. As the individual ages, it gets stronger and nut opening rate increases. Obviously the ape needs some knowledge, but we'll assume this comes easy and that body strength is the limiting factor. If the model does a poor job, then we'll have a good reason to reconsider this assumption.

In animals with terminal growth—they reach a stable adult body mass—size increases in proportion to the distance remaining to maximum size. This implies that the instantaneous rate of change in mass with age t is:

$$\frac{\mathrm{d}M}{\mathrm{d}t} = k(M_{\max} - M_t)$$

where k is a parameter that measures the rate of skill gain with age. The equation above tells us how fast mass changes at any given age. But we need a formula for the mass at a given age. Solving differential equations is beyond the level of this book. But you don't actually have to know how to solve it—any computer algebra system can do it. This particular differential equation is actually a biology classic,[224] and its solution is:

$$M_t = M_{\max}\big(1 - \exp(-kt)\big)$$

We'll plot this function later, when we do prior predictive simulations. It makes decelerating curves that level off at $M_{\max}$. If you want to glance ahead, examples are shown on the left in FIGURE 16.4 (page 552).

We actually care about strength. Mass isn't strength. So suppose now that strength is proportional to mass: $S_t = \beta M_t$. The parameter β simply measures the proportionality. Now we need some way to relate strength to the rate of nut cracking. We could assume it

too is simply proportional. But consider that strength helps in at least three ways. First, it let's the animal lift a heavier hammer. Heavier hammers have greater momentum. Second, it let's the animal accelerate the hammer faster than gravity. Third, stronger animals also have longer limbs, which gives them more efficient levers. So it makes sense to assume increasing returns to strength. Mathematically, this implies a function for the rate of nut opening like:

$$\lambda = \alpha S_t^\theta = \alpha\left(\beta M_{\max}(1 - \exp(-kt))\right)^\theta$$

where θ is some exponent greater than 1. A realistic implication of assuming increasing returns to strength is that there will be a threshold below which an individual cannot open a single nut in reasonable time. The new parameter α expresses the proportionality of strength to nut opening. It translates Newtons of force into nuts per second.

Now we have a function for the rate of nuts opened, λ. But it is a soup of parameters. We can simplify it, however. First, we can just rescale body mass $M_{\max}$ so that it equals 1. This might seem like cheating. But measurement scales are arbitrary. So making $M_{\max} = 1$ just sets the measurement scale. Doing this gives us:

$$\lambda = \alpha\beta^\theta(1 - \exp(-kt))^\theta$$

The product $\alpha\beta^\theta$ in the front just rescales strength to nuts-opened-per-second. So we can replace it with a single parameter:

$$\lambda = \phi(1 - \exp(-kt))^\theta$$

That's much better. One cost to this simplification is that it has hidden some useful facts. For example, average adult mass differs for males and females. An adult male chimpanzees can be 10 kilograms heavier than an adult female chimpanzee. You'll attempt to express that fact in a practice problem at the end of the chapter.

16.3.2. Statistical model. To use the model above for estimation, we need a likelihood function and priors. The likelihood is straightforward. If the number of nuts opened is far less than the number of available nuts, then the Poisson distribution has the right constraints. This gives us:

$$\begin{aligned} n_i &\sim \text{Poisson}(\lambda_i) \\ \lambda_i &= d_i\phi(1 - \exp(-kt_i))^\theta \end{aligned}$$

where n_i is the number of nuts opened, d_i is the duration spent opening nuts, and t_i is the individual's age on observation i. The only thing we've added is the exposure d_i. Back in Chapter 11, we added an exposure to a Poisson model by adding the log of the duration to the linear model. We don't use the log here, because the model isn't linear and has no log link function. We are coding the rate λ directly. So the duration d_i just multiplies the rate to give us the expected number of nuts opened. It is like if I told you that I can open $\lambda = 0.4$ nuts per second. To calculate how many nuts I could open in $d = 10$ seconds, you just multiply 0.4 by 10 to get 4 nuts per 10 seconds.

What about priors? To get sensible priors here, we need to consider relevant biological facts and also simulate to see how to translate those facts into distributional assumptions. The most relevant fact is that a chimpanzee reaches adult mass around 12 years of age. So the prior growth curves need to plateau around 12. We need distributions for k and θ that accomplish this. And then the prior for ϕ should have a mean around the maximum rate of nut opening. I am not really an expert on nut opening. But let's suppose a professional chimpanzee could open one nut per second—several nuts can be pounded at once.

Here are my suggestions for priors:

$$\phi \sim \text{Log-Normal}(\log(1), 0.1)$$
$$k \sim \text{Log-Normal}(\log(2), 0.25)$$
$$\theta \sim \text{Log-Normal}(\log(5), 0.25)$$

All three are Log-Normal, because all three parameters have to be positive and continuous. We can simulate from these priors and draw the implied prior growth and rate curves.

R code 16.10

```
N <- 1e4
phi <- rlnorm( N , log(1) , 0.1 )
k <- rlnorm( N , log(2), 0.25 )
theta <- rlnorm( N , log(5) , 0.25 )

# relative grow curve
plot( NULL , xlim=c(0,1.5) , ylim=c(0,1) , xaxt="n" , xlab="age" ,
    ylab="body mass" )
at <- c(0,0.25,0.5,0.75,1,1.25,1.5)
axis( 1 , at=at , labels=round(at*max(Panda_nuts$age)) )
for ( i in 1:20 ) curve( (1-exp(-k[i]*x)) , add=TRUE , col=grau() , lwd=1.5 )

# implied rate of nut opening curve
plot( NULL , xlim=c(0,1.5) , ylim=c(0,1.2) , xaxt="n" , xlab="age" ,
    ylab="nuts per second" )
at <- c(0,0.25,0.5,0.75,1,1.25,1.5)
axis( 1 , at=at , labels=round(at*max(Panda_nuts$age)) )
for ( i in 1:20 ) curve( phi[i]*(1-exp(-k[i]*x))^theta[i] , add=TRUE ,
    col=grau() , lwd=1.5 )
```

The plots are displayed in FIGURE 16.4. It will help to inspect the distribution of each parameter with `dens()`. But these plots that combine all of the parameters are essential for understanding their implications.

Coding this model presents no new problems. We just build the usual data list and express the likelihood and priors in `ulam()`:

R code 16.11

```
dat_list <- list(
    n = as.integer( Panda_nuts$nuts_opened ),
    age = Panda_nuts$age / max(Panda_nuts$age),
    seconds = Panda_nuts$seconds )

m16.4 <- ulam(
    alist(
        n ~ poisson( lambda ),
        lambda <- seconds*phi*(1-exp(-k*age))^theta,
        phi ~ lognormal( log(1) , 0.1 ),
        k ~ lognormal( log(2) , 0.25 ),
        theta ~ lognormal( log(5) , 0.25 )
    ), data=dat_list , chains=4 )
```

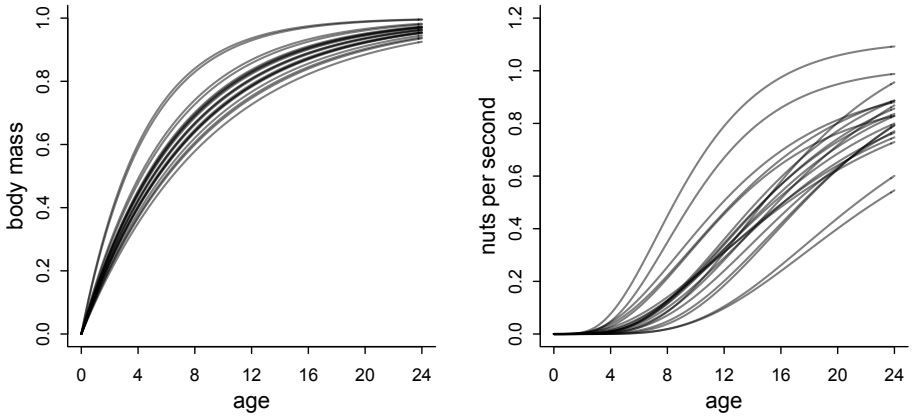


FIGURE 16.4. Prior predictive simulation for the nut opening model. Left: Prior growth curves, normalizing average adult mass to 1. This prior tries to start leveling off around age 12, like a real chimpanzee. Right: Prior nut opening rates. This prior allows many different developmental patterns. But they are all increasing with age and assume that baby chimpanzees cannot open nuts.

Now do your duty by checking the chain diagnostics. The marginal distribution of each parameter isn't as interesting here as the posterior developmental curve. So let's go straight to producing that.

R code 16.12

```
post <- extract.samples(m16.4)
plot( NULL , xlim=c(0,1) , ylim=c(0,1.5) , xlab="age" ,
    ylab="nuts per second" , xaxt="n" )
at <- c(0,0.25,0.5,0.75,1,1.25,1.5)
axis( 1 , at=at , labels=round(at*max(Panda_nuts$age)) )

# raw data
pts <- dat_list$n / dat_list$seconds
point_size <- normalize( dat_list$seconds )
points( jitter(dat_list$age) , pts , col=rangi2 , lwd=2 , cex=point_size*3 )

# 30 posterior curves
for ( i in 1:30 ) with( post ,
    curve( phi[i]*(1-exp(-k[i]*x))^theta[i] , add=TRUE , col=grau() ) )
```

The result is shown in FIGURE 16.5. The blue points are the raw data, with size scaled by the duration of each observation. The curves are 30 skill curves drawn from the posterior distribution. These curves level off around the age of maximum body size, consistent with the idea that strength is the main limiting factor. This doesn't mean that there isn't knowledge involved. There is still plenty of variation to explain.

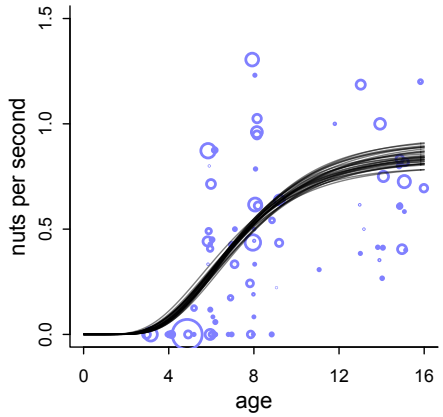


FIGURE 16.5. Posterior predictive distribution for the nut opening model. Blue points are raw data, number opened divided by seconds. Point size is proportional to the duration of that observation. The curves are 30 draws from the posterior distribution.

16.3.3. Covariates and individual differences. The model here is stupidly simple. But it is a scientifically reasonable start. You could extend it to include covariates like sex and individual differences in strength. There are repeat observations of individuals, and even repeat observations across different years, that could be used to estimate individual varying effects. The practice problems at the end of the chapter explore these applications.

Note for the moment that some of the model parameters make sense as varying by individual while others do not. The scaling parameter θ for example is a feature of the physics, not of an individual. Which parameters are allowed to vary by individual is something to be decided by scientific knowledge of the parameters. And this is another reason to avoid GLMs, so that the parameters have firmer scientific meaning.

Yet another improvement to this model might be to use a more realistic model of chimpanzee growth. There are detailed published growth curves for chimpanzees.[225] Male chimpanzees do experience a growth spurt around age 10. So their growth rate actually increases shortly before reaching maximum size. Incorporating this into the model might help improve predictions for males at least.

16.4. Population dynamics

It all starts with radiation released by fusion reactions inside a dwarf star in a minor arm of an insignificant spiral galaxy. Eight minutes away as the photon travels, on the third planet, that radiation allows clever plants to make sugar. Then the hare eats those clever plants and steals their sugar. The clever lynx eats the hare. Everyone is just eating starlight.

The population of hares and lynx fluctuate over time, and understanding nature requires understanding such fluctuations. The numbers of hares and lynx at any time influence the numbers in the near future. You might say that the most important cause of hares is hares. But predators, like the lynx, are also causes. To model phenomena like this, variables at one time influence the values of those same variables in the next.

In this section, we'll model a TIME SERIES of hare and lynx populations.[226] Load the data and display it:

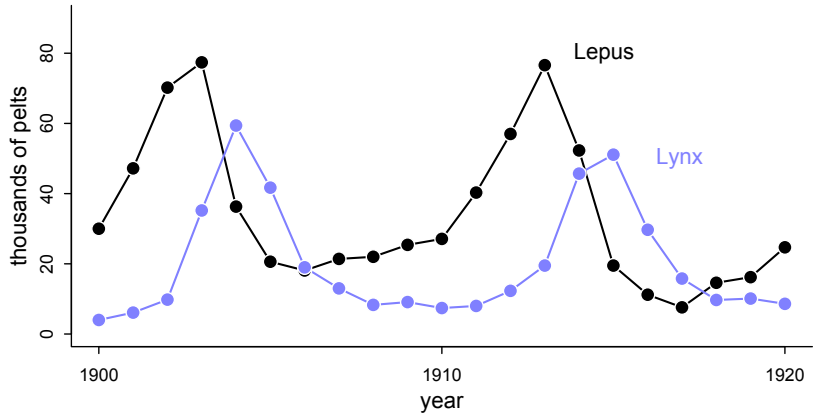


FIGURE 16.6. Twenty years of lynx (*Lynx canadensis*) and hare (*Lepus americanus*) pelts, as recorded by the Hudson Bay Company.

R code 16.13

```
library(rethinking)
data(Lynx_Hare)
plot( 1:21 , Lynx_Hare[,3] , ylim=c(0,90) , xlab="year" ,
    ylab="thousands of pelts" , xaxt="n" , type="l" , lwd=1.5 )
at <- c(1,11,21)
axis( 1 , at=at , labels=Lynx_Hare$Year[at] )
lines( 1:21 , Lynx_Hare[,2] , lwd=1.5 , col=rangi2 )
points( 1:21 , Lynx_Hare[,3] , bg="black" , col="white" , pch=21 , cex=1.4 )
points( 1:21 , Lynx_Hare[,2] , bg=rangi2 , col="white" , pch=21 , cex=1.4 )
text( 17 , 80 , "Lepus" , pos=2 )
text( 19 , 50 , "Lynx" , pos=2 , col=rangi2 )
```

FIGURE 16.6 displays this time series. These are odd data, records of pelts not live animals.[227] The number of hare pelts and number of lynx pelts seem to be related somehow. Both fluctuate, but they seem to fluctuate together.

A common geocentric way to model a time series like this would be to use something called an **AUTOREGRESSIVE MODEL**. In an autoregressive model, the value of the outcome in the previous time step is called a *lag variable* and added as a predictor to the right side of a linear model. For example, we might model the mean number of hares at time t as:

$$\mathrm{E}(H_t) = \alpha + \beta_1 H_{t-1}$$

where H_t is the number of hares at time t. If β_1 is less than 1, then hares tend to regress to some mean population size α. We could continue by adding an epicycle for the predator:

$$\mathrm{E}(H_t) = \alpha + \beta_1 H_{t-1} + \beta_2 L_{t-1}$$

where L_{t-1} is the number of lynx in the previous time period. Sometimes people add even deeper lags, like this:

$$\mathrm{E}(H_t) = \alpha + \beta_1 H_{t-1} + \beta_2 L_{t-1} + \beta_3 H_{t-2}$$

Now not only does the most recent population size H_{t-1} predict the present, but so too does the population size two time periods ago H_{t-2}. Everything from prices to temperature to wars has been modeled this way.

There are several famous problems with autoregressive models, despite how often they are used. They are surely generalized linear madness. First, nothing that happened two time periods ago causes the present, except through its influence on the state of the system one time period ago. So no lag beyond one period makes any causal sense. It's pure predictive epicycle. Of course some causal influences act slower than others. But that means you need another variable, not that the distance past can influence the present. Second, if the state of the system, H_t and L_t here, are measured with error, then the model is propagating error. It isn't the observed H_{t-1} that influences H_t, but rather the real unobserved H_{t-1}. In other words, what we really need is a **STATE SPACE MODEL**. Third, in most cases there is no biological, economic, or physical interpretation of the parameters. Consider for example the α intercept in the equations above. It implies that even when there are no hares, $H_{t-1} = 0$, there can be α hares in the next period. Sometimes all this nonsense is okay, if you care about is forecasting. But often these models don't even make good forecasts, because getting the future right often depends upon have a decent causal model.

It's easy to do better, if you use a little science. In this section, we'll model the hares and lynx using an incredibly basic ecological model. In the process, you'll see how to fit systems of **ORDINARY DIFFERENTIAL EQUATIONS** (ODEs) to data.

16.4.1. The scientific model. The hare population reproduces at a rate that depends upon the plants. And it shrinks at a rate that depends upon predators. Let H_t be the number of hares at time t. Then we can assert that the rate of change in the hare population is:

$$\frac{\mathrm{d}H}{\mathrm{d}t} = H_t(\text{birth rate}) - H_t(\text{death rate})$$

Everything is multiplied by H_t, because if there are no hares, then there are no births or deaths. Reproduction and death are *per capita* processes. The simplest ecological model makes birth and death rates constant. Let's call the birth rate b_H and the mortality rate m_H. This implies:

$$\frac{\mathrm{d}H}{\mathrm{d}t} = H_t b_H - H_t m_H = H_t(b_H - m_H)$$

The *per capita* growth rate is the difference between the birth rate and the death rate. I think of this as the first law of ecology. Every model must include it in some form.

The form we want to use here modifies the mortality rate so it also depends upon the presence of a predator, the clever lynx. Let L_t be the number of lynx at time t. Then we can write:

$$\frac{\mathrm{d}H}{\mathrm{d}t} = H_t(b_H - L_t m_H)$$

Similar logic gives us a similar equation for the rate of change in the lynx population:

$$\frac{\mathrm{d}L}{\mathrm{d}t} = L_t(H_t b_L - m_L)$$

In this case, it is birth that depends upon the other species and mortality that is a constant.

Now we have a model in which the population dynamics of the two species are determined by two coupled ordinary differential equations (ODEs). This isn't a realistic model. The plants that hares eat are not constantly available, and lynx eat more than just hares. But

let's see how far we can get with this model, a biological one in which the parameters mean something. Failures teach us.

This particular model is a famous one, the LOTKA-VOLTERRA MODEL.[228] It models simple predator-prey interactions and demonstrates several important things about ecological dynamics. Lots can be proved about it without using any data at all. For example, the population tends to be unstable, cycling up and down like in FIGURE 16.6. This is interesting, because it suggests that, while nature is more complicated, all that is necessary to see cyclical population dynamics is captured in a stupidly simple model.

The previous section also used a differential equation model. In that case we could explicitly solve it to get an expression for the value of the variable at any time t. We can't do that here. These equations have no explicit solution that tells us which H_t and L_t to expect at any time t. So how do we use them? We solve them numerically, through simulation. Let me show you how. Then we'll turn to making this into a statistical model.

A differential equation is just a way to update a variable. The equation $\mathrm{d}H/\mathrm{d}t$ tells us how to update H after each tiny unit of passing time $\mathrm{d}t$. This means that once we have a value for H, we can update it by just applying the equation $\mathrm{d}H/\mathrm{d}t$ over and over again. Specifically, we update like this:

$$H_{t+\mathrm{d}t} = H_t + \mathrm{d}t\frac{\mathrm{d}H}{\mathrm{d}t} = H_t + \mathrm{d}tH_t(b_H - L_t m_H)$$

We do have to be careful how we do this, because math in a computer is tricky, as you've seen before. In particular, the value we choose for $\mathrm{d}t$ needs to be small enough to provide a good approximation of continuous time. But this tactic really does work. And it allows us to see what the model implies, before we've fit it to data.

Let's write a function to simulate lynx-hare dynamics. This function just needs to apply the strategy above to both H and L. Here's some code that is hopefully easy to read:

R code 16.14

```
sim_lynx_hare <- function( n_steps , init , theta , dt=0.002 ) {
    L <- rep(NA,n_steps)
    H <- rep(NA,n_steps)
    L[1] <- init[1]
    H[1] <- init[2]
    for ( i in 2:n_steps ) {
        H[i] <- H[i-1] + dt*H[i-1]*( theta[1] - theta[2]*L[i-1] )
        L[i] <- L[i-1] + dt*L[i-1]*( theta[3]*H[i-1] - theta[4] )
    }
    return( cbind(L,H) )
}
```

We tell this function how long to simulate with `n_steps`, which initial population sizes to use with `init`, and which values of the parameters to use with `theta`. The time interval is `dt`. I've set it to default to 0.002, which works in this example. But the right value in general depends upon the model and the parameters.

Now let's use the function to simulate.

R code 16.15

```
theta <- c( 0.5 , 0.05 , 0.025 , 0.5 )
z <- sim_lynx_hare( 1e4 , as.numeric(Lynx_Hare[1,2:3]) , theta )
```

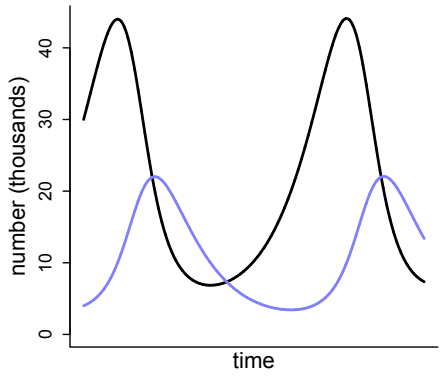


FIGURE 16.7. Simulated population dynamics from the lynx-hare model. Black: Hare population. Blue: Lynx population. This model produces repeating cycles of predators and prey.

```
plot( z[,2] , type="l" , ylim=c(0,max(z[,2])) , lwd=2 , xaxt="n" ,
    ylab="number (thousands)" , xlab="" )
lines( z[,1] , col=rangi2 , lwd=2 )
mtext( "time" , 1 )
```

The result is displayed in FIGURE 16.7. The black curve is the hare population, and the blue is the lynx population. This model produces cycles, similar to what we see in the data. The model behaves this way, because lynx eat hares. Once the hares are eaten, the lynx begin to die off. Then the cycle repeats.

16.4.2. The statistical model. To turn the lynx-hare model into a statistical analysis, we need to connect the deterministic population dynamics to the observed data. Observed data have many reasons not to exactly match a deterministic expectation. The most obvious is that we never get to count every hare and lynx. We just have partial samples. So we need to model both the underlying population dynamics and the observation process.

Let H_t and L_t as before represent the numbers of hares and lynx at time t. And now let h_t and ℓ_t represent the observed numbers of hares and lynx. While H_t causes $H_{t+\mathrm{d}t}$, the observed h_t does not cause anything. It's just a pale reflection of the unobserved state of the system at time t. We have to use a statistical model to project it back to the underly model of H_t and L_t. Then we can make a prediction for $h_{t+\mathrm{d}t}$ and $\ell_{t+\mathrm{d}t}$.

To do this, we need to assign an error distribution to the observation process. To do this in a principled way, we should outline the generative process that goes from the true state of nature, H_t, to the measurement, h_t. First, hares get trapped. Suppose each hare is trapped with some probability p_t which varies year to year, for all sorts of reasons. Third, the actual number of pelts were rounded to the nearest 100 and divided by 1000. So they are no longer counts exactly. This all sounds like a mess. That's measurement for you.

We can do this though. Suppose for example there is a population of $H_t = 10^4$ hares. Suppose also that the annual trapping rate varies according to a beta distribution $p_t \sim \mathrm{Beta}(2, 18)$. This means the average is 10%, but it is very rarely double that. We get a binomial count of pelts sampled for the population of hares, and then that is rounded to the nearest 100 and divided by 1000. Let's see what this sort of distribution looks like:

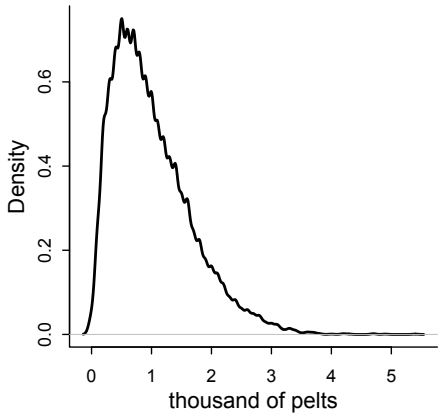


FIGURE 16.8. Simulated distribution for the observation model in which trapping probability varies from year to year. In this case, a wide range of pelt counts are consistent with the same true population size. This makes inference about population size difficult.

R code 16.16

```
N <- 1e4
Ht <- 1e4
p <- rbeta(N,2,18)
h <- rbinom( N , size=Ht , prob=p )
h <- round( h/1000 , 2 )
dens( h , xlab="thousand of pelts" , lwd=2 )
```

I show this density in FIGURE 16.8. The variation in p_t leads to a skewed error distribution. Try changing `Ht` and the distribution of `p` in the code above and see how the distribution changes.

There are several reasonable ways to approximate this distribution. We could for example just use a Log-Normal distribution. It has the right constraints and skew. For example, we could assign:

$$h_t \sim \text{Log-Normal}(\log(pH_t), \sigma_H)$$

This gives h_t a median of pH_t, the expected proportion of the hare population that is trapped. The parameter σ_H controls the dispersion. An important fact about this measurement process is that there is no good way to estimate p, not without lots of data at least. So we're going to just fix it with a strong prior. If this makes you uncomfortable, notice that the model has forced us to realize that we cannot do any better than relative population estimates, unless we have a good way to know p. A typical time series model would just happily spin on its epicycles, teaching us nothing this useful.

We will ignore rounding error, since it is at most $50/4000 = 0.0125 = 1.25\%$ of the pelt count. But if you are curious how to incorporate rounding into a statistical model, see the Overthinking box later on. It isn't hard to do—we just think generatively and that provides the solution.

Let's lay out the full statistical model now. First we have the probabilities of the observed variables, the pelts:

$$h_t \sim \text{Log-Normal}(\log(p_H H_t), \sigma_H) \quad \text{[Prob observed hare pelts]}$$
$$\ell_t \sim \text{Log-Normal}(\log(p_L L_t), \sigma_L) \quad \text{[Prob observed lynx pelts]}$$

Then we need to define the unobserved variables. Let's start with the unobserved population sizes of lynx L_t and hare H_t.

$$
\begin{aligned}
H_1 &\sim \text{Log-Normal}(\log 10, 1) && \text{[Prior for initial hare population]} \\
L_1 &\sim \text{Log-Normal}(\log 10, 1) && \text{[Prior for initial lynx population]} \\
H_{T>1} &= H_1 + \int_1^T H_t(b_H - m_H L_t)\mathrm{d}t && \text{[Model for hare population]} \\
L_{T>1} &= L_1 + \int_1^T L_t(b_L H_t - m_L)\mathrm{d}t && \text{[Model for lynx population]}
\end{aligned}
$$

The first two lines above assign priors to the initial population sizes at time $t = 1$. The differential equation model then, in the third and fourth lines, defines all times after that. And finally all the parameters need priors.

$$
\begin{aligned}
\sigma_H &\sim \text{Exponential}(1) && \text{[Prior for measurement dispersion]} \\
\sigma_L &\sim \text{Exponential}(1) && \text{[Prior for measurement dispersion]} \\
p_H &\sim \text{Beta}(\alpha_H, \beta_H) && \text{[Prior for hare trap probability]} \\
p_L &\sim \text{Beta}(\alpha_L, \beta_L) && \text{[Prior for lynx trap probability]} \\
b_H &\sim \text{Half-Normal}(1, 0.5) && \text{[Prior hare birth rate]} \\
b_L &\sim \text{Half-Normal}(0.05, 0.05) && \text{[Prior lynx birth rate]} \\
m_H &\sim \text{Half-Normal}(0.05, 0.05) && \text{[Prior hare mortality rate]} \\
m_L &\sim \text{Half-Normal}(1, 0.5) && \text{[Prior lynx mortality rate]}
\end{aligned}
$$

In the problems at the end of the chapter, I'll ask you to conduct prior predictive simulations with these priors.

Now we're ready to start engineering the sampler. The obstacle in this model is computing H_t and L_t for each time t. The differential equations define these variables, but our sampler need to numerically solve them on each iteration. So we need to write the numerical integration we did earlier, when we simulated the model, into our Bayesian sampler. Fortunately, Stan already has functions for solving differential equations. So this will be easier than it sounds. The Stan User's Guide (https://mc-stan.org) contains a full section on programming this type of model, with several examples.

We'll do this model directly in Stan. You can load the Stan code and display it with:

R code 16.17

```
data(Lynx_Hare_model)
cat(Lynx_Hare_model)
```

I won't reproduce the entire model here. But I will point out the unusual pieces that handle the differential equations. The first unusual piece is at the top, the `functions` block. This is an optional block that lets us write special calculations that we can use in the model. This is where we put a function that computes the values of the differential equations. Look at the code—seriously, look at it—and you'll see the `dpop_dt` function at the start of the model. The `pop` here is for population. This function returns the rates of change in the population. It takes as input the time `t`, the initial state of the population `pop_init`, and a vector of parameters `theta`. Then it computes the rates of change in lynx and hares.

The model uses this function to determine the values of H_t and L_t. All we really have to do is pass the name of the function and its inputs to Stan's helpful `integrate_ode_rk45` function. This function does the integration for us. In this model, we do this in the `transformed parameters` block, so the results will appear as parameters in the posterior. But they are actually deterministic functions of the other parameters, the birth and mortality rates and the initial population sizes. The results are stored in a matrix called `pop`, which has a row for each observed time point and a column for each species.

The rest of the model is rather ordinary. The `model` block declares the priors and relates the solved equations to the observed data with:

```
  for ( t in 1:N )
    for ( k in 1:2 )
      pelts[t,k] ~ lognormal( log(pop[t,k]*p[k]) , sigma[k] );
```

There is also code in `generated quantities` to go ahead and perform posterior predictive simulations. We'll plot those after sampling.

Now we're ready. Prepare the data list and fire up the engines:

R code 16.18

```
dat_list <- list(
    N = nrow(Lynx_Hare),
    pelts = Lynx_Hare[,2:3] )

m16.5 <- stan( model_code=Lynx_Hare_model , data=dat_list , chains=3 , cores=3 ,
    control=list( adapt_delta=0.95 ) )
```

As always, check the chains. But sampling should be rapid and smooth. You could inspect the parameters. Each has a biological meaning. But they all cooperate in a very non-linear way to produce the population dynamics, so it isn't easy to read the dynamics from the individual parameters. So let's plot posterior predictions, at both the pelt (observed) and population (unobserved) levels. For the pelts, this will plot the raw data and overly 21 simulated time series from the posterior.

R code 16.19

```
post <- extract.samples(m16.5)
pelts <- dat_list$pelts
plot( 1:21 , pelts[,2] , pch=16 , ylim=c(0,120) , xlab="year" ,
    ylab="thousands of pelts" , xaxt="n" )
at <- c(1,11,21)
axis( 1 , at=at , labels=Lynx_Hare$Year[at] )
points( 1:21 , pelts[,1] , col=rangi2 , pch=16 )
# 21 time series from posterior
for ( s in 1:21 ) {
    lines( 1:21 , post$pelts_pred[s,,2] , col=col.alpha("black",0.2) , lwd=2 )
    lines( 1:21 , post$pelts_pred[s,,1] , col=col.alpha(rangi2,0.3) , lwd=2 )
}
# text labels
text( 17 , 90 , "Lepus" , pos=2 )
text( 19 , 50 , "Lynx" , pos=2 , col=rangi2 )
```

The result is shown in the top plot of FIGURE 16.9. The black points and trends are the hare pelts. The blue points and trends are the lynx pelts. Note the jaggedness of the predicted trends. This is a result of the model assuming uncorrelated measurement errors across time

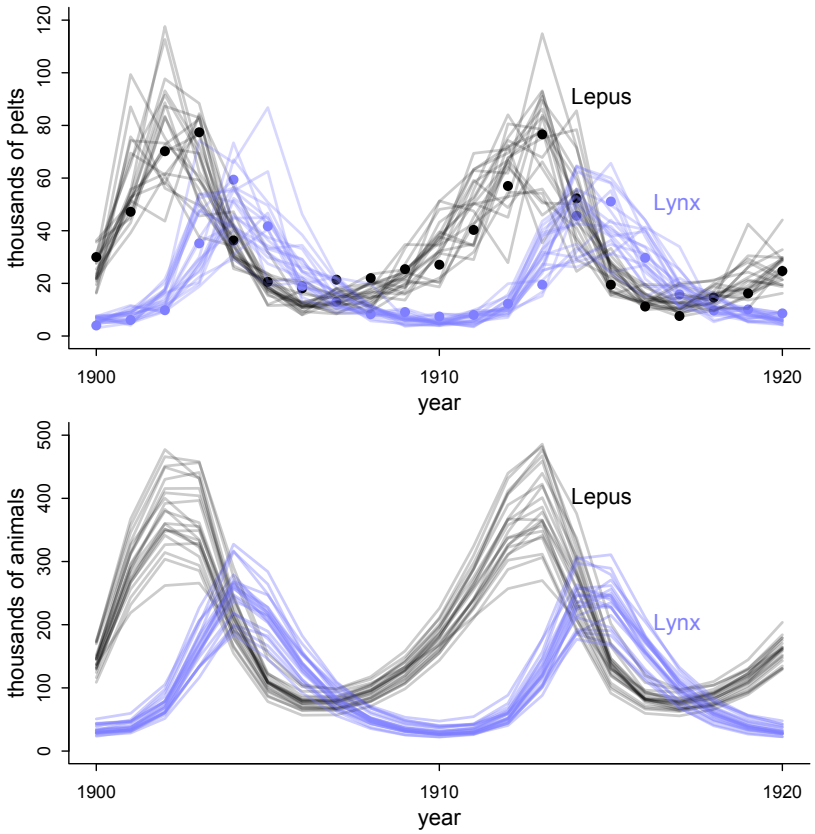


FIGURE 16.9. Posterior predictions for the lynx-hare model. Top: Posterior pelts. The points are the data, black for hares and blue for lynx. Each trend is a predicted time series from the posterior distribution. The jagged path is caused by uncorrelated measurement error. Bottom: Posterior populations. Unlike the pelt predictions, these are smooth trajectories without measurement error.

points. The underlying population may be smooth, but the measurements will not be. This is an example of why it is almost always a mistake to model a time series as if observed data cause observed data in the next time step. This is what is often done in autoregressive models. But if there is measurement error, and there always is, the data are emissions of some unseen state. The hidden states are the causes. The measurements don't cause anything.

It is helpful to compare the pelt predictions to the population predictions. So here are 21 simulations of population dynamics from the posterior:

R code 16.20

```
plot( NULL , pch=16 , xlim=c(1,21) , ylim=c(0,500) , xlab="year" ,
    ylab="thousands of animals" , xaxt="n" )
at <- c(1,11,21)
```

```
axis( 1 , at=at , labels=Lynx_Hare$Year[at] )
for ( s in 1:21 ) {
    lines( 1:21 , post$pop[s,,2] , col=col.alpha("black",0.2) , lwd=2 )
    lines( 1:21 , post$pop[s,,1] , col=col.alpha(rangi2,0.4) , lwd=2 )
}
```

The result is the bottom plot in FIGURE 16.9. Compared to the pelt time series, these population dynamics are smooth. There is a lot of uncertainty about population size, of course. But each trajectory connects smoothly, because there is no measurement error at this level. The differential equation model is deterministic, so it shows no stochasticity.

16.4.3. Lynx lessons. There are good reasons to doubt that this model is a perfect explanation of the population dynamics of hares and lynx. While lynx really do depend almost exclusively on hares at times, hares are eaten by lots of predators. So the hare cycles are probably not caused by the lynx. In other words, there is a confound lurking here. Real ecologies are complicated. In the practice problems at the end, I'll ask you to use this model on an experimental predator-prey system that lacks all those complexities. I'll also ask you to compare an autoregressive model and see how many epicycles you need to approach the forecasting quality of the simple predator-prey model.

16.5. Summary

This chapter demonstrated four analyses in which a statistical model is motived directly by a scientific model. This approach stands in contrast to the customary approach of going directly from a vague scientific model, whether a DAG or just a bowl of variables, to a generalized linear model. The goal was to illustrate both the advantages and challenges of translating scientifically informed structural causal models into statistical machines. The goal was not to persuade you to never use a generalized linear model. But hopefully it inspires you to see the use of a GLM as a decision in itself, not an obligation.

16.6. Practice

Easy.

16E1. x

Medium.

16M1. Modify the cylinder height model, `m16.1`, so that the exponent 3 on height is instead a free parameter. Do you recover the value of 3 or not? Plot the posterior predictions for the new model. How do they differ from those of `m16.1`?

16M2. Conduct a prior predictive simulation for the cylinder height model. Begin with the priors in the chapter. Do these produce reasonable prior height distributions? If not, which modifications do you suggest?

16M3. Use prior predictive simulations to investigate the Lynx-hare model. Begin with the priors in the chapter. Which population dynamics do these produce? Can you suggest any improvements to the priors, on the basis of your simulations?

Hard.