6 The Haunted DAG & The Causal Terror

It seems like the most newsworthy scientific studies are the least trustworthy. The more likely it is to kill you, if true, the less likely it is to be true. The more boring the topic, the more rigorous the results. How could this widely believed negative correlation exist? There doesn't seem to be any reason for studies of topics that people care about to produce less reliable results. Maybe popular topics attract more and worse researchers, like flies drawn to the smell of honey?

Actually all that is necessary for such a negative correlation to arise is that peer reviewers care about both newsworthiness and trustworthiness. Whether it is grant review or journal review, if editors and reviewers care about both, then the act of selection itself is enough to make the most newsworthy studies the least trustworthy. In fact, it's hard to imagine how scientific peer review could avoid creating this negative correlation. And, dear reader, this fact will help us understand the perils of multiple regression.

Here's a simple simulation to illustrate the point[85]. Suppose a grant review panel receives 200 proposals for scientific research. Among these proposals, there is no correlation at all between trustworthiness (rigor, scholarship, plausibility of success) and newsworthiness (social welfare value, public interest). The panel weighs trustworthiness and newsworthiness equally. Then they rank the proposals by their combined scores and select the top 10% for funding.

At the end of this section, I show the code to simulate this thought experiment. Figure 6.1 displays the full sample of simulated proposals, with those selected in blue. I've drawn a simple linear regression line through the selected proposals. There's the negative correlation, -0.77 in this example. Strong selection induces a negative correlation among the criteria used in selection. Why? If the only way to cross the threshold is to score high, it is more common to score high on one item than on both. Therefore among funded proposals, the most newsworthy studies can actually have less than average trustworthiness (less than 0 in the figure). Similarly the most trustworthy studies can actually be less newsworthy than average.

This general phenomenon has been recognized for a long time. It is sometimes called Berkson's paradox[86]. But it is easier to remember if we call it the *selection-distortion effect*. Once you appreciate this effect, you'll see it everywhere. Why do so many restaurants in good locations have bad food? The only way a restaurant with less-than-good food can survive is if it is in a nice location. Similarly, restaurants with excellent food can survive even in bad locations. Selection-distortion ruins your city.

What does this have to do with multiple regression? Unfortunately, everything. The previous chapter demonstrated some amazing powers of multiple regression. It can smoke

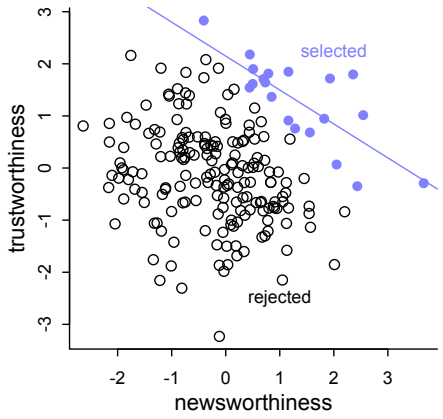


FIGURE 6.1. Why the most newsworthy studies might be the least trustworthy. 200 research proposals are ranked by combined trustworthiness and newsworthiness. The top 10% are selected for funding. While there is no correlation before selection, the two criteria are strongly negatively correlated after selection. The correlation here is −0.77.

out spurious correlations and clear up masking effects. This may encourage the view that, when in doubt, just add everything to the model and let the oracle of regression sort it out.

Regression will not sort it out. Regression is indeed an oracle, but a cruel one. It speaks in riddles and delights in punishing us for asking bad questions. The selection-distortion effect can happen inside of a multiple regression, because the act of adding a predictor induces statistical selection within the model, a phenomenon that goes by the unhelpful name COLLIDER BIAS. This can mislead us into believing, for example, that there is a negative association between newsworthiness and trustworthiness in general, when in fact it is just a consequence of conditioning on some variable. This is both a deeply confusing fact and one that is important to understand in order to regress responsibly.

This chapter and the next are both about terrible things that can happen when we simply add variables to a regression, without a clear idea of a causal model. In this chapter, we'll explore three different hazards: multicollinearity, post-treatment bias, and collider bias. We'll end by tying all of these examples together in a framework that can tell us which variables we must and must not add to a model in order to arrive at valid inferences. But this framework does not do the most important step for us: It will not give us a valid model.

Overthinking: Simulated science distortion. Simulations like this one are easy to do in R, or in any other scripting language, once you have seen a few examples. In this simulation, we just draw some random Gaussian criteria for a sample of proposals and then select the top 10% combined scores.

R code 6.1

```
set.seed(1914)
N <- 200 # num grant proposals
p <- 0.1 # proportion to select
# uncorrelated newsworthiness and trustworthiness
nw <- rnorm(N)
tw <- rnorm(N)
# select top 10% of combined scores
s <- nw + tw  # total score
q <- quantile( s , 1-p ) # top 10% threshold
selected <- ifelse( s >= q , TRUE , FALSE )
cor( tw[selected] , nw[selected] )
```

I chose a specific seed so you can replicate the result in FIGURE 6.1, but if you rerun the simulation without the `set.seed` line, you'll see there is nothing special about the seed I used.

6.1. Multicollinearity

It is commonly true that there are many potential predictor variables to add to a regression model. In the case of the primate milk data, for example, there are 7 variables available to predict any column we choose as an outcome. Why not just fit a model that includes all 7? There are several hazards. The one we'll focus on here is **MULTICOLLINEARITY**. Multicollinearity means very strong correlation between two or more predictor variables. The consequence of it is that the posterior distribution will seem to suggest that none of the variables is reliably associated with the outcome, even if all of the variables are in reality strongly associated with the outcome. This frustrating phenomenon arises from the details of how multiple regression works. In fact, there is nothing wrong with multicollinearity. The model will work fine for prediction. You will just be frustrated trying to understand it. The hope is that once you understand multicollinearity, you will better understand regression models in general.

Let's begin with a simple simulation. Then we'll turn to the primate milk data again and see multicollinearity in a real data set.

6.1.1. Multicollinear legs. The simulation example is predicting an individual's height using the length of his or her legs as predictor variables. Surely height is positively associated with leg length, or at least the simulation will assume it is. Nevertheless, once you put both leg lengths into the model, something vexing will happen.

The code below will simulate the heights and leg lengths of 100 individuals. For each, first a height is simulated from a Gaussian distribution. Then each individual gets a simulated proportion of height for their legs, ranging from 0.4 to 0.5. Finally, each leg is salted with a little measurement or developmental error, so the left and right legs are not exactly the same length, as is typical in real populations. At the end, the code puts height and the two leg lengths into a common data frame.

R code 6.2

```
N <- 100                          # number of individuals
set.seed(909)
height <- rnorm(N,10,2)           # sim total height of each
leg_prop <- runif(N,0.4,0.5)      # leg as proportion of height
leg_left <- leg_prop*height +     # sim left leg as proportion + error
    rnorm( N , 0 , 0.02 )
leg_right <- leg_prop*height +    # sim right leg as proportion + error
    rnorm( N , 0 , 0.02 )
                                  # combine into data frame
d <- data.frame(height,leg_left,leg_right)
```

Now let's analyze these data, predicting the outcome `height` with both predictors, `leg_left` and `leg_right`. Before approximating the posterior, however, consider what we expect. On average, an individual's legs are 45% of their height (in these simulated data). So we should expect the beta coefficient that measures the association of a leg with height to end up around the average height (10) divided by 45% of the average height (4.5). This is $10/4.5 \approx 2.2$. Now

let's see what happens instead. I'll use very vague, bad priors here, just so we can be sure that the priors aren't responsible for what is about to happen.

R code 6.3

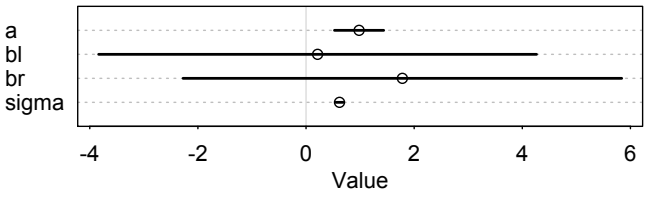
```
m6.1 <- quap(
    alist(
        height ~ dnorm( mu , sigma ) ,
        mu <- a + bl*leg_left + br*leg_right ,
        a ~ dnorm( 10 , 100 ) ,
        bl ~ dnorm( 2 , 10 ) ,
        br ~ dnorm( 2 , 10 ) ,
        sigma ~ dexp( 1 )
    ) ,
    data=d )
precis(m6.1)
```

```
      mean   sd  5.5% 94.5%
a     0.98 0.28  0.53  1.44
bl    0.21 2.53 -3.83  4.25
br    1.78 2.53 -2.26  5.83
sigma 0.62 0.04  0.55  0.69
```

Those posterior means and standard deviations look crazy. This is a case in which a graphical view of the `precis` output is more useful, because it displays the posterior means and 89% intervals in a way that allows us with a glance to see that something has gone wrong here:

R code 6.4

```
plot(precis(m6.1))
```



Go ahead and try the simulate a few more times, omitting the `set.seed` line. If both legs have almost identical lengths, and height is so strongly associated with leg length, then why is this posterior distribution so weird? Did the posterior approximation work correctly?

It did work correctly, and the posterior distribution here is the right answer to the question we asked. The problem is the question. Recall that a multiple linear regression answers the question: *What is the value of knowing each predictor, after already knowing all of the other predictors?* So in this case, the question becomes: *What is the value of knowing each leg's length, after already knowing the other leg's length?*

The answer to this weird question is equally weird, but perfectly logical. The posterior distribution is the answer to this question, considering every possible combination of the parameters and assigning relative plausibilities to every combination, conditional on this model and these data. It might help to look at the bivariate posterior distribution for `bl` and `br`:

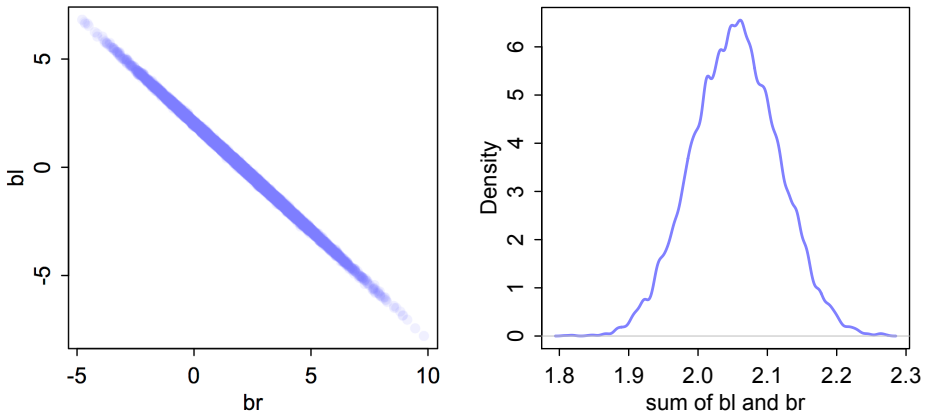


FIGURE 6.2. Left: Posterior distribution of the association of each leg with height, from model `m6.1`. Since both variables contain almost identical information, the posterior is a narrow ridge of negatively correlated values. Right: The posterior distribution of the sum of the two parameters is centered on the proper association of either leg with height.

R code 6.5

```
post <- extract.samples(m6.1)
plot( bl ~ br , post , col=col.alpha(rangi2,0.1) , pch=16 )
```

The resulting plot is shown on the left of FIGURE 6.2. The posterior distribution for these two parameters is very highly correlated, with all of the plausible values of `bl` and `br` lying along a narrow ridge. When `bl` is large, then `br` must be small. What has happened here is that since both leg variables contain almost exactly the same information, if you insist on including both in a model, then there will be a practically infinite number of combinations of `bl` and `br` that produce the same predictions.

One way to think of this phenomenon is that you have approximated this model:

$$y_i \sim \text{Normal}(\mu_i, \sigma)$$
$$\mu_i = \alpha + \beta_1 x_i + \beta_2 x_i$$

The variable y is the outcome, like height in the example, and x is a single predictor, like the leg lengths in the example. Here x is used twice, which is a perfect example of the problem caused by using the almost-identical leg lengths. From the computer's perspective, this model is simply:

$$y_i \sim \text{Normal}(\mu_i, \sigma)$$
$$\mu_i = \alpha + (\beta_1 + \beta_2) x_i$$

All I've done is factor x_i out of each term. The parameters β_1 and β_2 cannot be pulled apart, because they never separately influence the mean μ. Only their sum, $\beta_1 + \beta_2$, influences μ. So this means the posterior distribution ends up reporting the very large range of combinations of β_1 and β_2 that make their sum close to the actual association of x with y.

And the posterior distribution in this simulated example has done exactly that: It has produced a good estimate of the sum of `bl` and `br`. Here's how you can compute the posterior distribution of their sum, and then plot it:

R code 6.6

```
sum_blbr <- post$bl + post$br
dens( sum_blbr , col=rangi2 , lwd=2 , xlab="sum of bl and br" )
```

And the resulting density plot is shown on the right-hand side of FIGURE 6.2. The posterior mean is in the right neighborhood, a little over 2, and the standard deviation is much smaller than it is for either component of the sum, `bl` or `br`. If you fit a regression with only one of the leg length variables, you'll get approximately the same posterior mean:

R code 6.7

```
m6.2 <- quap(
    alist(
        height ~ dnorm( mu , sigma ) ,
        mu <- a + bl*leg_left,
        a ~ dnorm( 10 , 100 ) ,
        bl ~ dnorm( 2 , 10 ) ,
        sigma ~ dexp( 1 )
    ) ,
    data=d )
precis(m6.2)
```

```
      mean   sd 5.5% 94.5%
a     1.00 0.28 0.54  1.45
bl    1.99 0.06 1.89  2.09
sigma 0.62 0.04 0.55  0.69
```

That 1.99 is almost identical to the mean value of `sum_blbr`.

You'll get slightly different results in your own simulation, due to random variation across simulations. But the basic lesson remains intact across different simulations: *When two predictor variables are very strongly correlated, including both in a model may lead to confusion*. The posterior distribution isn't wrong, in such cases. It's telling you that the question you asked cannot be answered with these data. And that's a great thing for a model to say, that it cannot answer your question. And if you are just interested in prediction, you'll find that this leg model makes fine predictions. It just doesn't make any claims about which leg is more important.

This leg example is clear and cute. But it is also purely statistical. We aren't asking any serious causal questions here. Let's try a more causally interesting example next.

6.1.2. Multicollinear milk.

In the leg length example, it's easy to see that including both legs in the model is a little silly. But the problem that arises in real data sets is that we may not anticipate a clash between highly correlated predictors. And therefore we may mistakenly read the posterior distribution to say that neither predictor is important. In this section, we look at an example of this issue with real data.

Let's return to the primate milk data from earlier in the chapter. Let's get back the original data again:

R code 6.8

```
library(rethinking)
data(milk)
d <- milk
d$K <- scale( d$kcal.per.g )
d$F <- scale( d$perc.fat )
d$L <- scale( d$perc.lactose )
```

In this example, we are concerned with the variables `perc.fat` (percent fat) and `perc.lactose` (percent lactose) that we might use to model the total energy content, `kcal.per.g`. The code above has already standardized these three variables. You're going to use these three variables to explore a natural case of multicollinearity. Note that there are no missing values, `NA`, in these columns, so there's no need here to extract complete cases. But you can rest assured that `quap`, unlike reckless functions like `lm`, would never silently drop cases.

Start by modeling `kcal.per.g` as a function of `perc.fat` and `perc.lactose`, but in two bivariate regressions. Look back in Chapter 5 (page 151), for a discussion of these priors.

R code 6.9

```
# kcal.per.g regressed on perc.fat
m6.3 <- quap(
    alist(
        K ~ dnorm( mu , sigma ) ,
        mu <- a + bF*F ,
        a ~ dnorm( 0 , 0.2 ) ,
        bF ~ dnorm( 0 , 0.5 ) ,
        sigma ~ dexp( 1 )
    ) , data=d )

# kcal.per.g regressed on perc.lactose
m6.4 <- quap(
    alist(
        K ~ dnorm( mu , sigma ) ,
        mu <- a + bL*L ,
        a ~ dnorm( 0 , 0.2 ) ,
        bL ~ dnorm( 0 , 0.5 ) ,
        sigma ~ dexp( 1 )
    ) , data=d )

precis( m6.3 )
precis( m6.4 )
```

```
      mean   sd  5.5% 94.5%
a     0.00 0.08 -0.12  0.12
bF    0.86 0.08  0.73  1.00
sigma 0.45 0.06  0.36  0.54

       mean   sd  5.5% 94.5%
a      0.00 0.07 -0.11  0.11
bL    -0.90 0.07 -1.02 -0.79
sigma  0.38 0.05  0.30  0.46
```

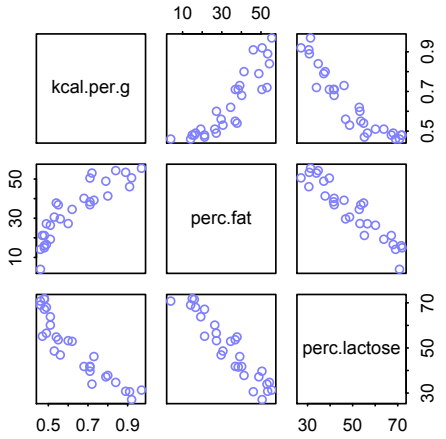


FIGURE 6.3. A pairs plot of the total energy, percent fat, and percent lactose variables from the primate milk data. Percent fat and percent lactose are strongly negatively correlated with one another, providing mostly the same information.

The posterior distributions for `bF` and `bL` are essentially mirror images of one another. The posterior mean of `bF` is as positive as the mean of `bL` is negative. Both are narrow posterior distributions that lie almost entirely on one side or the other of zero. Given the strong association of each predictor with the outcome, we might conclude that both variables are reliable predictors of total energy in milk, across species. The more fat, the more kilocalories in the milk. The more lactose, the fewer kilocalories in milk. But watch what happens when we place both predictor variables in the same regression model:

R code 6.10

```
m6.5 <- quap(
    alist(
        K ~ dnorm( mu , sigma ) ,
        mu <- a + bF*F + bL*L ,
        a ~ dnorm( 0 , 0.2 ) ,
        bF ~ dnorm( 0 , 0.5 ) ,
        bL ~ dnorm( 0 , 0.5 ) ,
        sigma ~ dexp( 1 )
    ) ,
    data=d )
precis( m6.5 )
```

```
       mean   sd  5.5% 94.5%
a      0.00 0.07 -0.11  0.11
bF     0.24 0.18 -0.05  0.54
bL    -0.68 0.18 -0.97 -0.38
sigma  0.38 0.05  0.30  0.46
```

Now the posterior means of both `bF` and `bL` are closer to zero. And the standard deviations for both parameters are twice as large as in the bivariate models (`m6.3` and `m6.4`).

This is the same statistical phenomenon as in the leg length example. What has happened is that the variables `perc.fat` and `perc.lactose` contain much of the same information.

They are almost substitutes for one another. As a result, when you include both in a regression, the posterior distribution ends up describing a long ridge of combinations of `bF` and `bL` that are equally plausible. In the case of the fat and lactose, these two variables form essentially a single axis of variation. The easiest way to see this is to use a pairs plot:

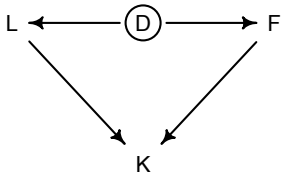
```
pairs( ~ kcal.per.g + perc.fat + perc.lactose , data=d , col=rangi2 )
```

R code 6.11

I display this plot in FIGURE 6.3. Along the diagonal, the variables are labeled. In each scatterplot off the diagonal, the vertical axis variable is the variable labeled on the same row and the horizontal axis variable is the variable labeled in the same column. For example, the two scatterplots in the first row in FIGURE 6.3 are `kcal.per.g` (vertical) against `perc.fat` (horizontal) and then `kcal.per.g` (vertical) against `perc.lactose` (horizontal). Notice that percent fat is positively correlated with the outcome, while percent lactose is negatively correlated with it. Now look at the right-most scatterplot in the middle row. This plot is the scatter of percent fat (vertical) against percent lactose (horizontal). Notice that the points line up almost entirely along a straight line. These two variables are negatively correlated, and so strongly so that they are nearly redundant. Either helps in predicting `kcal.per.g`, but neither helps much *once you already know the other*.

In the scientific literature, you might encounter a variety of dodgy ways of coping with multicollinearity. Few of them take a causal perspective. Some fields actually teach students to inspect pairwise correlations before fitting a model, to identify and drop highly correlated predictors. This is a mistake. Pairwise correlations are not the problem. It is the conditional associations—not correlations—that matter. And even then, the right thing to do will depend upon what is causing the collinearity. The associations within the data alone are not enough to decide what to do.

What is likely going on in the milk example is that there is a core tradeoff in milk composition that mammal mothers must obey. If a species nurses often, then the milk tends to be watery and low in energy. Such milk is high in sugar (lactose). If instead a species nurses rarely, in short bouts, then the milk needs to be higher in energy. Such milk is very high in fat. This implies a causal model something like this:



The central tradeoff decides how dense, D, the milk needs to be. We haven't observed this variable, so it's shown circled. Then fat, F, and lactose, L, are determined. Finally, the composition of F and L determines the kilocalories, K. If we could measure D, or had an evolutionary and economic model to predict it based upon other aspects of a species, that would be better than stumbling through regressions. We'd just regression K on D, ignoring the mediating L and F, to estimate the causal influence of density on energy.

The problem of multicollinearity is a member of a family of problems with fitting models, a family sometimes known as **NON-IDENTIFIABILITY**. When a parameter is non-identifiable, it means that the structure of the data and model do not make it possible to estimate the parameter's value. Sometimes this problem arises from mistakes in coding a model, but many important types of models present non-identifiable or weakly identifiable parameters,

even when coded completely correctly. Nature does not owe us easy inference, even when the model is correct.

In general, there's no guarantee that the available data contain much information about a parameter of interest. When that's true, your Bayesian machine will return a posterior distribution very similar to the prior. Comparing the posterior to the prior can therefore be a good idea, a way of seeing how much information the model extracted from the data. When the posterior and prior are similar, it doesn't mean the calculations are wrong—you got the right answer to the question you asked. But it might lead you to ask a better question.

Rethinking: Identification guaranteed; comprehension up to you. Technically speaking, *identifiability* is not a concern for Bayesian models. The reason is that as long as the posterior distribution is proper—which just means that it integrates to 1—then all of the parameters are identified. But this technical fact doesn't also mean that you can make sense of the posterior distribution. So it's probably better to speak of *weakly identified* parameters in a Bayesian context. There will be several examples as the book progresses.

Overthinking: Simulating collinearity. The code to produce FIGURE **??** involves writing a function that generates correlated predictors, fits a model, and returns the standard deviation of the posterior distribution for the slope relating `perc.fat` to `kcal.per.g`. Then the code repeatedly calls this function, with different degrees of correlation as input, and collects the results.

R code 6.12

```
library(rethinking)
data(milk)
d <- milk
sim.coll <- function( r=0.9 ) {
    d$x <- rnorm( nrow(d) , mean=r*d$perc.fat ,
        sd=sqrt( (1-r^2)*var(d$perc.fat) ) )
    m <- lm( kcal.per.g ~ perc.fat + x , data=d )
    sqrt( diag( vcov(m) ) )[2] # stddev of parameter
}
rep.sim.coll <- function( r=0.9 , n=100 ) {
    stddev <- replicate( n , sim.coll(r) )
    mean(stddev)
}
r.seq <- seq(from=0,to=0.99,by=0.01)
stddev <- sapply( r.seq , function(z) rep.sim.coll(r=z,n=100) )
plot( stddev ~ r.seq , type="l" , col=rangi2, lwd=2 , xlab="correlation" )
```

So for each correlation value in `r.seq`, the code generates 100 regressions and returns the average standard deviation from them. This code uses implicit flat priors, which are bad priors. So it does exaggerate the effect of collinear variables. When you use informative priors, the inflation in standard deviation can be much slower.

6.2. Post-treatment bias

It is routine to worry about mistaken inferences that arise from omitting predictor variables. Such mistakes are often called **OMITTED VARIABLE BIAS**, and the examples from the previous chapter illustrate it. It is much less routine to worry about mistaken inferences arising from including variables that are consequences of other variables. We'll call this **POST-TREATMENT BIAS**.[87] Being aware of post-treatment bias is important in all types of studies.

Carefully controlled experiments can be ruined just as easily as uncontrolled observational studies. Blindly tossing variables into the causal salad is never a good idea, no matter how the data were collected.

The language "post-treatment" comes in fact from thinking about experimental designs. Suppose for example that you are growing some plants in a greenhouse. You want to know the difference in growth under different anti-fungal soil treatments, because fungus on the plants tends to reduce their growth. Plants are initially seeded and sprout. Their heights are measured. Then different soil treatments are applied. Final measures are the height of the plant and the presence of fungus. There are four variables of interest here: initial height, final height, treatment, and presence of fungus. Final height is the outcome of interest. But which of the other variables should be in the model? If your goal is to make a causal inference about the treatment, you shouldn't include the presence of fungus, because it is a *post-treatment* effect.

Let's simulate some data, to make the example more transparent and see what exactly goes wrong when we include a post-treatment variable.

R code 6.13

```
set.seed(71)
# number of plants
N <- 100

# simulate initial heights
h0 <- rnorm(N,10,2)

# assign treatments and simulate fungus and growth
treatment <- rep( 0:1 , each=N/2 )
fungus <- rbinom( N , size=1 , prob=0.5 - treatment*0.4 )
h1 <- h0 + rnorm(N, 5 - 3*fungus)

# compose a clean data frame
d <- data.frame( h0=h0 , h1=h1 , treatment=treatment , fungus=fungus )
precis(d)
```

```
           mean   sd  5.5% 94.5%    histogram
h0         9.96 2.10  6.57 13.08 ▁▁▃▇▃▂▁▁▁▁
h1        14.40 2.69 10.62 17.93      ▁▂▅▇▇▃▁
treatment  0.50 0.50  0.00  1.00   ▇▁▁▁▁▁▁▁▇
fungus     0.23 0.42  0.00  1.00   ▇▁▁▁▁▁▁▁▂
```

Now you should have a data frame `d` with the simulated plant experiment data.

6.2.1. A prior is born. When designing the model, it helps to pretend you don't have the data generating process just above. In real research, you will not know the real data generating process. But you will have a lot of scientific information to guide model construction. So let's spend some time taking this mock analysis seriously.

We know that the plants at time $t = 1$ should be taller than at time $t = 0$, whatever scale they are measured on. So if we put the parameters on a scale of *proportion* of height at time $t = 0$, rather than on the absolute scale of the data, we can set the priors more easily. To make this simpler, let's focus right now only on the height variables, ignoring the predictor

variables. We might have a linear model like:

$$h_{1,i} \sim \text{Normal}(\mu_i, \sigma)$$
$$\mu_i = h_{0,i} \times p$$

where $h_{0,i}$ is plant i's height at time $t = 0$, $h_{1,i}$ is its height at time $t = 1$, and p is a parameter measuring the proportion of $h_{0,i}$ that $h_{1,i}$ is. More precisely, $p = h_{1,i}/h_{0,i}$. If $p = 1$, the plant hasn't changed at all from time $t = 0$ to time $t = 1$. If $p = 2$, it has doubled in height. So if we center our prior for p on 1, that implies an expectation of no change in height. That is less than we know. But we should allow p to be less than 1, in case the experiment goes horribly wrong and we kill all the plants. We also have to ensure that $p > 0$, because it is a proportion. Back in Chapter 4 (page 4.4.1.3), we used a Log-Normal distribution, because it is always positive. Let's use one again. If we use $p \sim \text{Log-Normal}(0, 0.25)$, the prior distribution looks like:

R code 6.14

```
sim_p <- rlnorm( 1e4 , 0 , 0.25 )
precis( data.frame(sim_p) )
```

```
'data.frame': 10000 obs. of 1 variables:
      mean   sd 5.5% 94.5%   histogram
sim_p 1.03 0.26 0.67  1.48 ▁▃▇▃▁▁▁▁▁
```

So this prior expects anything from 40% shrinkage up to 50% growth. Let's fit this model, so you can see how it just measures the average growth in the experiment.

R code 6.15

```
m6.6 <- quap(
    alist(
        h1 ~ dnorm( mu , sigma ),
        mu <- h0*p,
        p ~ dlnorm( 0 , 0.25 ),
        sigma ~ dexp( 1 )
    ), data=d )
precis(m6.6)
```

```
      mean   sd 5.5% 94.5%
p     1.43 0.02 1.40  1.45
sigma 1.79 0.13 1.59  1.99
```

About 40% growth, on average. Now to include the treatment and fungus variables. We'll include both of them, following the notion that we'd like to measure the impact of both the treatment and the fungus itself. The parameters for these variables will also be on the proportion scale. They will be *changes* in proportion growth. So we're going to make a linear

model of p now.

$$\begin{aligned} h_{1,i} &\sim \text{Normal}(\mu_i, \sigma) \\ \mu_i &= h_{0,i} \times p \\ p &= \alpha + \beta_T T_i + \beta_F F_i \\ \alpha &\sim \text{Log-Normal}(0, 0.25) \\ \beta_T &\sim \text{Normal}(0, 0.5) \\ \beta_F &\sim \text{Normal}(0, 0.5) \\ \sigma &\sim \text{Exponential}(1) \end{aligned}$$

The proportion of growth p is now a function of the predictor variables. It looks like any other linear model. The priors on the slopes are almost certainly too flat. They place 95% of the prior mass between -1 (100% reduction) and $+1$ (100% increase) and two-thirds of the prior mass between -0.5 and $+0.5$. After we finish this section, you may want to loop back and try simulating from these priors. Here's the code to approximate the posterior:

R code 6.16

```
m6.7 <- quap(
    alist(
        h1 ~ dnorm( mu , sigma ),
        mu <- h0 * p,
        p <- a + bt*treatment + bf*fungus,
        a ~ dlnorm( 0 , 0.2 ) ,
        bt ~ dnorm( 0 , 0.5 ),
        bf ~ dnorm( 0 , 0.5 ),
        sigma ~ dexp( 1 )
    ), data=d )
precis(m6.7)
```

```
       mean   sd  5.5% 94.5%
a      1.48 0.02  1.44  1.52
bt     0.00 0.03 -0.05  0.05
bf    -0.27 0.04 -0.33 -0.21
sigma  1.41 0.10  1.25  1.57
```

That `a` parameter is the same as `p` before. And it has nearly the same posterior. The marginal posterior for `bt`, the effect of treatment, is solidly zero, with a tight interval. The treatment is not associated with growth. The fungus seems to have hurt growth, however. Given that we know the treatment matters, because we built the simulation that way, what happened here?

6.2.2. Blocked by consequence. The problem is that `fungus` is mostly a consequence of `treatment`. This is to say that `fungus` is a post-treatment variable. So when we control for `fungus`, the model is implicitly answering the question: *Once we already know whether or not a plant developed fungus, does soil treatment matter?* The answer is "no," because soil treatment has its effects on growth through reducing fungus. But we actually want to know, based on the design of the experiment, is the impact of treatment on growth. To measure this properly, we should omit the post-treatment variable `fungus`. Here's what the inference looks like in that case:

R code 6.17

```
m6.8 <- quap(
    alist(
        h1 ~ dnorm( mu , sigma ),
        mu <- h0 * p,
        p <- a + bt*treatment,
        a ~ dlnorm( 0 , 0.2 ),
        bt ~ dnorm( 0 , 0.5 ),
        sigma ~ dexp( 1 )
    ), data=d )
precis(m6.8)
```

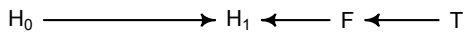
```
      mean   sd 5.5% 94.5%
a     1.38 0.03 1.34  1.42
bt    0.08 0.03 0.03  0.14
sigma 1.75 0.12 1.55  1.94
```

Now the impact of treatment is clearly positive, as it should be. It makes sense to control for pre-treatment differences, like the initial height `h0`, that might mask the causal influence of treatment. But including post-treatment variables can actually mask the treatment itself. This doesn't mean you don't want the model that includes both treatment and fungus. The fact that including `fungus` zeros the coefficient for `treatment` suggests that the treatment works for exactly the anticipated reasons. It tells us about mechanism. But a correct inference about the treatment still depends upon omitting the post-treatment variable.

6.2.3. Fungus and *d*-separation. It helps to look at this problem in terms of a DAG. In this case, I'll show you how to draw it using the `dagitty` R package, because we are going to use that package now to do some graph analysis.

R code 6.18

```
library(dagitty)
plant_dag <- dagitty( "dag {
    H_0 -> H_1
    F -> H_1
    T -> F
}")
coordinates( plant_dag ) <- list( x=c(H_0=0,T=2,F=1.5,H_1=1) ,
                                  y=c(H_0=0,T=0,F=0,H_1=0) )
drawdag( plant_dag )
```



So the treatment T influences the presence of fungus F which influences plant height at time 1, H_1. Plant height at time 1 is also influenced by plant height at time 0, H_0. That's our DAG. When we include F, the post-treatment effect, in the model, we end up blocking the path from the treatment to the outcome. This is the DAG way of saying that learning the treatment tells us nothing about the outcome, once we know the fungus status.

An even more DAG way to say this is that conditioning on F induces **D-SEPARATION**. The "d" stands for *directional*.[88] *d*-separation means that some variables on a directed graph are independent of others. There is no path connecting them. In this case, H_1 is *d*-separated from

T, but only when we condition on F. Conditioning on F effectively blocks the directed path $T \rightarrow F \rightarrow H_1$, making T and H_1 independent (d-separated). In the previous chapter, you saw the notation $H_1 \perp\!\!\!\perp T|F$ for this kind of statement, when we discussed implied **CONDITIONAL INDEPENDENCIES**. Why does this happen? There is no information in T about H_1 that is not also in F. So once we know F, learning T provides no additional information about H_1. You can query the implied conditional independencies for this DAG:

R code 6.19

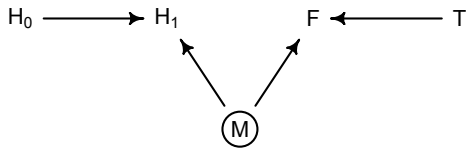
```
impliedConditionalIndependencies(plant_dag)
```

```
F _||_ H0
H0 _||_ T
H1 _||_ T | F
```

There are three. The third one is the focus of our discussion. But the other two implications provide ways to test the DAG. What $F \perp\!\!\!\perp H_0$ and $H_0 \perp\!\!\!\perp T$ say is that the original plant height, H_0, should not be associated with the treatment T or fungus F, provided we do not condition on anything.

Obviously the problem of post-treatment variables applies just as well to observational studies as it does to experiments. But in experiments, it can be easier to tell which variables are pre-treatment, like `h0`, and which are post-treatment, like `fungus`. In observational studies, it is harder to know. But there are some subtle traps in experiments as well.

For example, conditioning on a post-treatment variable can not only fool you into thinking the treatment doesn't work. It can also fool you into thinking it does work. Consider the DAG below:



In this graph, the treatment T influences fungus F, but fungus doesn't influence plant growth. Maybe the plant species just isn't bothered by this particular fungus. The new variable M is moisture. It influences both H_1 and F. M is circled to indicate that it is unobserved. Any unobserved common cause of H_1 and F will do—it doesn't have to be moisture of course. A regression of H_1 on T will show no association between the treatment and plant growth. But if we include F in the model, suddenly there will be an association. Let's try it. I'll just modify the plant growth simulation so that fungus has no influence on growth, but moisture M influences both H_1 and F:

R code 6.20

```
set.seed(71)
N <- 1000
h0 <- rnorm(N,10,2)
treatment <- rep( 0:1 , each=N/2 )
M <- rbern(N)
fungus <- rbinom( N , size=1 , prob=0.5 - treatment*0.4 + 0.4*M )
h1 <- h0 + rnorm( N , 5 + 3*M )
d2 <- data.frame( h0=h0 , h1=h1 , treatment=treatment , fungus=fungus )
```

Rerun the models from earlier, models `m6.7` and `m6.8`, using the data in `d2` now. You'll see that including fungus again confounds inference about the treatment, this time by making it seem like it helped the plants, even though it had no effect.

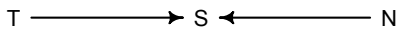
This result is rather mysterious. Why should M have this effect? The next section is all about effects like this.

Rethinking: Model selection doesn't help. In the next chapter, you'll learn about model selection using information criteria. Like other model comparison and selection schemes, these criteria help in contrasting and choosing model structure. But such approaches are no help in the example presented just above, since the model that includes `fungus` both fits the sample better and would make better out-of-sample predictions. Model `m6.7` misleads because it asks the wrong question, not because it would make poor predictions. As argued in Chapter 1, prediction and causal inference are just not the same task. No statistical procedure can substitute for scientific knowledge and attention to it. We need multiple models because they help us understand causal paths, not just so we can choose one or another for prediction.

6.3. Collider bias

At the start of the chapter, I argued that all that is necessary for scientific studies to show a negative association between trustworthiness and newsworthiness is that selection processes—grant and journal review—care about both. Now I want to explain how this same selection phenomenon can happen inside a statistical model. When it does, it can seriously distort our inferences, a phenomenon known as **COLLIDER BIAS**.

Let's consider a DAG for this example. The model is that trustworthiness (T) and newsworthiness (N) are statistically independent in the population research proposals submitted to grant review panels. Both of them influence selection (S) for funding. This is the graph:



The fact that two arrows enter S means it is a **COLLIDER**. The core concept is easy to understand: When you condition on a collider, it creates statistical—but not necessarily causal—associations among its causes. In this case, once you learn that a proposal has been selected (S), then learning its trustworthiness (T) also provides information about its newsworthiness (N). Why? Because if, for example, a selected proposal has low trustworthiness, then it must have high newsworthiness. Otherwise it wouldn't have been funded. The same works in reverse: If a proposal has low newsworthiness, we'd infer that it must have higher than average trustworthiness. Otherwise it would not have been selected for funding.

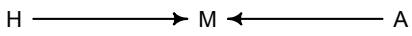
This is the informational phenomenon that generates the negative association between T and N in the population of selected proposals. And it means we have to pay attention to processes that select our sample of observations and may distort associations among variables. But the same phenomenon will also generate a misleading association inside a statistical model, when you include the collider as a predictor variable. If you are not careful, you can make an erroneous causal inference. Let's consider an extended example.

6.3.1. Collider of false sorrow.

Consider the question of how aging influences happiness. If we have a large survey of people rating how happy they are, is age associated with happiness?

If so, is that association causal? Here, I want to show you how controlling for a plausible confound of happiness can actually bias inference about the influence of age.[89]

Suppose, just to be provocative, that an individual's average happiness is a trait that is determined at birth and does not change with age. However, happiness does influence events in one's life. One of those events is marriage. Happier people are more likely to get married. Another variable that causally influences marriage is age: The more years you are alive, the more likely you are to eventually get married. Putting these three variables together, this is the causal model:



Happiness (H) and age (A) both cause marriage (M). Marriage is therefore a collider. Even though there is no causal association between happiness and age, if we condition on marriage—which means here, if we include it as a predictor in a regression—then it will induce a statistical association between age and happiness. And this can mislead us to think that happiness changes with age, when in fact it is constant.

To convince you of this, let's do another simulation. Simulations are useful in these examples, because these are the only times when we know the true causal model. If a procedure cannot figure out the truth in a simulated example, we shouldn't trust it in a real one. We're going to do a fancier simulation this time, using an agent-based model of aging and marriage to produce a simulated data set to use in a regression. Here is the simulation design:

(1) Each year, 20 people are born with uniformly distributed happiness values.
(2) Each year, each person ages one year. Happiness does not change.
(3) At age 18, individuals can become married. The odds of marriage each year are proportional to an individual's happiness.
(4) Once married, an individual remains married.
(5) After age 65, individuals leave the sample. (They move to Spain.)

I've written this algorithm into the `rethinking` package. You can run it out for 1000 years and collect the resulting data:

R code 6.21

```
library(rethinking)
d <- sim_happiness( seed=1977 , N_years=1000 )
precis(d)
```

```
'data.frame': 1300 obs. of 3 variables:
          mean    sd  5.5% 94.5%     histogram
age       33.0 18.77  4.00 62.00 ▇▇▇▇▇▇▇▇▇▇▇▇
married    0.3  0.46  0.00  1.00 ▇▁▁▁▁▁▁▁▁▂
happiness  0.0  1.21 -1.79  1.79    ▂▃▅▇▇▅▃▂
```

These data comprise 1300 people of all ages from birth to 65 years old. The variables correspond to the variables in the DAG above, and the simulation itself obeys the DAG.

I've plotted these data in Figure 6.4, showing each individual as a point. Filled points are married individuals. Age is on the horizontal, and happiness the vertical, with the happiest individuals at the top. At age 18, they become able to marry, and then gradually more individuals are married each year. So at older ages, more individuals are married. But at all ages, the happiest individuals are more likely to be married.

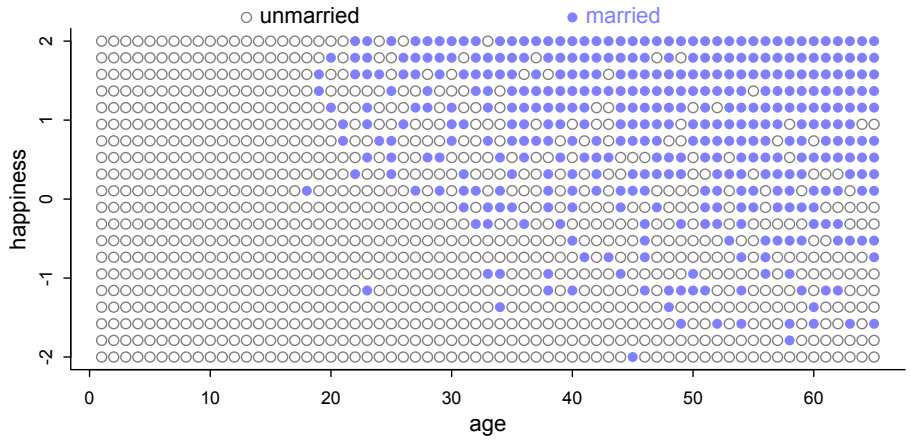


FIGURE 6.4. Simulated data, assuming that happiness is uniformly distributed and never changes. Each point is a person. Married individuals are shown with filled blue points. At each age after 18, the happiest individuals are more likely to be married. At later ages, more individuals tend to be married. Marriage status is a collider of age and happiness: A $\rightarrow$ M $\leftarrow$ H. If we condition on marriage in a regression, it will mislead us to believe that happiness declines with age.

Suppose you come across these data and want to ask whether age is related to happiness. You don't know the true causal model. But you reason, reasonably, that marriage status might be an important confound. If married people are more or less happy, on average, then you need to condition on marriage status in order to infer the relationship between age and happiness.

So let's consider a multiple regression model aimed at inferring the influence of age on happiness, while controlling for marriage status. This is just a plain multiple regression, like the others in this and the previous chapter. The linear model is this:

$$\mu_i = \alpha_{\text{MID}[i]} + \beta_A A_i$$

where $\text{MID}[i]$ is an index for the marriage status of individual i, with 1 meaning single and 2 meaning married. This is just the categorical variable strategy from Chapter 4. It's easier to make priors, when we use multiple intercepts, one for each category, than when we use indicator variables.

Now we should do our duty and think about the priors. Let's consider the slope β_A first, because how we scale the predictor A will determine the meaning of the intercept. We'll focus only on the adult sample, those 18 or over. Imagine a very strong relationship between age and happiness, such that happiness is at its maximum at age 18 and its minimum at age 65. It'll be easier if we rescale age so that the range from 18 to 65 is one unit. This will do it:

R code 6.22

```
d2 <- d[ d$age>17 , ] # only adults
d2$A <- ( d2$age - 18 ) / ( 65 - 18 )
```

Now this new variable A ranges from 0 to 1, where 0 is age 18 and 1 is age 65. Happiness is on an arbitrary scale, in these data, from -2 to $+2$. So our imaginary strongest relationship, taking happiness from maximum to minimum, has a slope with rise over run of $(2-(-2))/1 = 4$. Remember that 95% of the mass of a normal distribution is contained within 2 standard deviations. So if we set the standard deviation of the prior to half of 4, we are saying that we expect 95% of plausible slopes to be less than maximally strong. That isn't a very strong prior, but again, it at least helps bound inference to realistic ranges. Now for the intercepts. Each α is the value of μ_i when $A_i = 0$. In this case, that means at age 18. So we need to allow α to cover the full range of happiness scores. Normal$(0, 1)$ will put 95% of the mass in the -2 to $+2$ interval.

Finally, let's approximate the posterior. We need to construct the marriage status index variable, as well. I'll do that, and then immediate present the quap code.

R code 6.23

```
d2$mid <- d2$married + 1
m6.9 <- quap(
    alist(
        happiness ~ dnorm( mu , sigma ),
        mu <- a[mid] + bA*A,
        a[mid] ~ dnorm( 0 , 1 ),
        bA ~ dnorm( 0 , 2 ),
        sigma ~ dexp(1)
    ) , data=d2 )
precis(m6.9,depth=2)
```

```
       mean   sd  5.5% 94.5%
a[1]  -0.23 0.06 -0.34 -0.13
a[2]   1.26 0.08  1.12  1.40
bA    -0.75 0.11 -0.93 -0.57
sigma  0.99 0.02  0.95  1.03
```

The model is quite sure that age is negatively associated with happiness. We'd like to compare the inferences from this model to a model that omits marriage status. Here it is, followed by a comparison of the marginal posterior distributions:

R code 6.24

```
m6.10 <- quap(
    alist(
        happiness ~ dnorm( mu , sigma ),
        mu <- a + bA*A,
        a ~ dnorm( 0 , 1 ),
        bA ~ dnorm( 0 , 2 ),
        sigma ~ dexp(1)
    ) , data=d2 )
precis(m6.10)
```

```
      mean   sd  5.5% 94.5%
a     0.00 0.08 -0.12  0.12
bA    0.00 0.13 -0.21  0.21
sigma 1.21 0.03  1.17  1.26
```

This model, in contrast, finds no association between age and happiness.

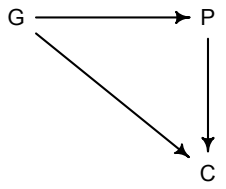
The pattern above is exactly what we should expect when we condition on a collider. The collider is marriage status. It a common consequence of age and happiness. As a result, when we condition on it, we induce a spurious association between the two causes. So it looks like, to model `m6.9`, that age is negatively associated with happiness. But this is just a statistical association, not a causal association. Once we know whether someone is married or not, then their age does provide information about how happy they are.

You can see this in FIGURE 6.4. Consider only the blue points, the married people. Among only the blue points, older individuals have lower average happiness. This is because more people get marriage at time goes on, so the mean happiness among married people approaches the population average of zero. Now consider only the open points, the unmarried people. Here it is also true that mean happiness declines with age. This is because happier individuals migrate over time into the married sub-population. So in both the married and unmarried sub-populations, there is a negative relationship between age and happiness. But in neither sub-population does this accurately reflect causation.

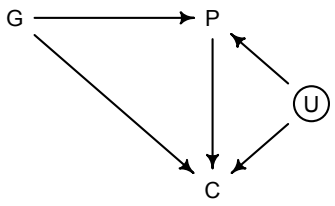
It's easy to plead with this example. Shouldn't marriage also influence happiness? What if happiness does change with age? But this misses the point. If you don't have a causal model, you can't make inferences from a multiple regression. And the regression itself does not provide the evidence you need to justify a causal model. Instead, you need some science.

6.3.2. The haunted DAG. Collider bias arises from conditioning on a common consequence, as in the previous example. If we can just get our graph sorted, we can avoid it. But it isn't always so easy to see a potential collider, because there may be unmeasured causes. Unmeasured causes can still induce collider bias. So I'm sorry to say that we also have to consider the possibility that our DAG may be haunted.

Suppose for example that we are interested in inferring the direct influence of both parents (P) and grandparents (G) on the educational achievement of children (C).[90] Since grandparents also presumably influence their own children's education, there is an arrow $G \rightarrow P$. This sounds pretty easy, so far. It's similar in structure to our divorce rate example from last chapter:



But suppose there are unmeasured, common influences on parents and their children, such as neighborhoods, that are not shared by grandparents (who live on the south coast of Spain now). Then our DAG becomes haunted by the unobserved U:



Now P is a common consequence of G and U, so if we condition on P, it will bias inference about $G \rightarrow C$, *even if we never get to measure* U. I don't expect that fact to be immediately obvious. So let's crawl through a quantitative example.

First, let's simulate 200 triads of grandparents, parents, and children. This simulation will be simple. We'll just project our DAG as a series of implied functional relationships. The DAG above implies that:

(1) P is some function of G and U
(2) C is some function of G, P, and U
(3) G and U are not functions of any other known variables

We can make these implications into a simple simulation, using `rnorm` to generate simulated observations. But to do this, we need to be a bit more precise than "some function of." So I'll invent some strength of association:

R code 6.25

```
N <- 200  # number of grandparent-parent-child triads
b_GP <- 1 # direct effect of G on P
b_GC <- 0 # direct effect of G on C
b_PC <- 1 # direct effect of P on C
b_U <- 2  # direct effect of U on P and C
```

These parameters are like slopes in a regression model. Notice that I've assumed that grandparents G have zero effect on their grandkids C. The example doesn't depend upon that effect being exactly zero, but it will make the lesson clearer. Now we use these slopes to draw random observations:

R code 6.26

```
set.seed(1)
U <- 2*rbern( N , 0.5 ) - 1
G <- rnorm( N )
P <- rnorm( N , b_GP*G + b_U*U )
C <- rnorm( N , b_PC*P + b_GC*G + b_U*U )
d <- data.frame( C=C , P=P , G=G , U=U )
```

I've made the neighborhood effect, U, binary. This will make the example easier to understand. But the example doesn't depend upon that assumption. The other lines are just linear models embedded in `rnorm`.

Now what happens when we try to infer the influence of grandparents? Since some of the total effect of grandparents passes through parents, we realize we need to control for parents. Here is a simple regression of C on P and G. Normally I would advise standardizing the variables, because it makes establishing sensible priors a lot easier. But I'm going to keep the simulated data on its original scale, so you can see what happens to inference about the slopes above. If we changed the scale, we shouldn't expect to get those values back. But if we leave the scale alone, we should be able to recover something close to those values. So I apologize for using vague priors here, just to push forward in the example.

R code 6.27

```
m6.11 <- quap(
    alist(
        C ~ dnorm( mu , sigma ),
        mu <- a + b_PC*P + b_GC*G,
```

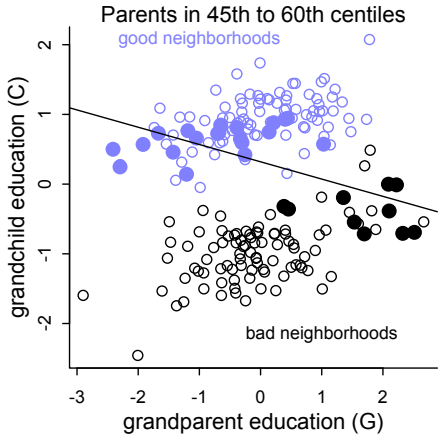


FIGURE 6.5. Unobserved confounds and collider bias. In this example, grandparents influence grandkids only indirectly, through parents. However, unobserved neighborhood effects on parents and their children create the illusion that grandparents harm their grandkids education. Parental education is a collider: Once we condition on it, grandparental education becomes negatively associated with grandchild education.

```
        a ~ dnorm( 0 , 1 ),
        c(b_PC,b_GC) ~ dnorm( 0 , 1 ),
        sigma ~ dexp( 1 )
    ), data=d )
precis(m6.11)
```

```
       mean   sd  5.5% 94.5%
a     -0.12 0.10 -0.28  0.04
b_PC   1.79 0.04  1.72  1.86
b_GC  -0.84 0.11 -1.01 -0.67
sigma  1.41 0.07  1.30  1.52
```

The inferred effect of parents looks too big, almost twice as large as it should be. That isn't surprising. Some of the correlation between P and C is due to U, and the model doesn't know about U. That's a simple confound. More surprising is that the model is confident that the direct effect of grandparents is to hurt their grandkids. The regression is not wrong. But a causal interpretation of that association would be.

How does collider bias arise in this case? Consider FIGURE 6.5. Note that I did standardize the variables to make this plot. So the units on the axes are standard deviations. The horizontal axis is grandparent education. The vertical is grandchild education. There are two clouds of points. The blue cloud comprises children who live in good neighborhoods ($U = 1$). The black cloud comprises children who live in bad neighborhoods ($U = -1$). Notice that both clouds of points show positive associations between G and C. More educated grandparents have more educated grandkids, but this effect arises entirely through parents. Why? Because we assumed it is so. The direct effect of G in the simulation is zero.

So how does the negative association arise, when we condition on parents? Conditioning on parents is like looking within sub-populations of parents with similar education. So let's try that. In FIGURE 6.5, I've highlighted in filled points those parents between the 45th and 60th centiles of education. There is nothing special of this range. It just makes the phenomenon easier to see. Now if we draw a regression line through only these points, regressing C on G, the slope is negative. There is the negative association that our multiple regression finds. But why does it exist?