8. Chapter 8 Solutions

8E1. There are many good responses. Here are some examples.

(1) Bread dough rises because of yeast, conditional on the presence of sugar. This is an interaction between yeast and sugar.
(2) Education leads to higher income, conditional on field of study. Some types of degrees lead to higher income than others.
(3) Gasoline makes a car go, unless it has no spark plugs (or belts, or wheels, etc.).

8E2. Only number (1) is a strict interaction. The others all imply additive influences. For (1), you could express it as "cooking with low heat leads to caramelizing, conditional on the onions not drying out." That's an interaction, as the effect of heat depends upon moisture. For (2), a car with many cylinders can go fast, whether or not it also has a good fuel injector. The reverse is also implied: either a fuel injector or more cylinders is sufficient to make a car go faster. For (3), the statement does not imply that the influence of parents depends upon the beliefs of friends. It just implies that one may be influenced by either parents or friends. For (4), the word "or" gives away that this is another case in which either factor, sociality or manipulative appendages (hands, tentacles), is sufficient to predict intelligence.

8E3.

(1) For outcome "extent caramelized", $\mu_i = \alpha + \beta_H H_i + \beta_M M_i + \beta_{HM} H_i M_i$.
(2) For outcome "maximum speed", $\mu_i = \alpha + \beta_C C_i + \beta_F F_i$.
(3) For outcome "extent conservative", $\mu_i = \alpha + \beta_P P_i + \beta_F F_i$.
(4) For outcome "intelligence" (whatever that means when comparing an octopus to a monkey), $\mu_i = \alpha + \beta_S S_i + \beta_M M_i$.

8M1. Tulips are a winter flower in much of their natural range. High temperatures do frustrate them. This is not a *linear* interaction, because by raising temperature the effect was to prevent all blooms. A linear effect would be an additive change. But it is still correct to say that there is a three-way interaction here: the influence of water and shade depend upon one another, and both and their interaction depend upon temperature. In later chapters, you'll see how to model non-linear responses of this kind.

8M2. Here is one idea. Let L_i stand for the ordinary linear model from the chapter. Then let H_i be a 0/1 indicator of whether or not the temperature was hot. Then:

$$\mu_i = L_i(1 - H_i)$$

When $H_i = 1$, the entire model above is zero, regardless of the value of L_i.

8M3. The implied relationship between ravens and wolves is one in which wolves do not need ravens, but ravens very much do benefit from wolves (at least in some places). Here's an example set of data in which this might be the case:

Region	Wolves	Ravens
1	12	43
2	15	46
3	7	28
4	30	99
5	17	60
6	70	212

Really, this "interaction" is not a statical interaction effect at all, because just stating that ravens depend upon wolves implies that we can partially predict raven density with wolf density. A statistical interaction requires instead that some other third variable regulate the dependency of ravens on wolves. For example, in regions in which there is plenty of small prey for ravens to kill and consume on their own, the presence of wolves may not matter at all.

8M4. The direct approach is to use log-normal or exponential distributions. Both of these distributions are strictly positive. We can force the shade effect to be negative by using a minus in the linear model.

Thinking about the interaction is harder. But our prior information tells us that more water increases the effect of light. So similarly more water decreases the impact of shade. So the interaction should be negative. So we can force the interaction effect to also have a log-normal or exponential, but make it negative by adding a minus in the linear model.

Here's the code with log-normal. Remember (or look up) that the mean of a log-normal is $\exp(\mu + \sigma^2/2)$. So we can't make these values very large at all, if we want to prior mean to be reasonable. Let's load the data and try a first guess:

R code 8.1

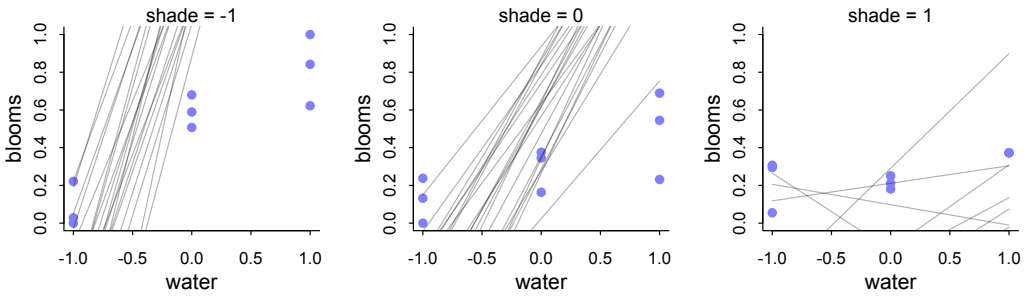
```
library(rethinking)
data(tulips)
d <- tulips
d$blooms_std <- d$blooms / max(d$blooms)
d$water_cent <- d$water - mean(d$water)
d$shade_cent <- d$shade - mean(d$shade)

m8M4a <- quap(
    alist(
        blooms_std ~ dnorm( mu , sigma ) ,
        mu <- a + bw*water_cent - bs*shade_cent - bws*water_cent*shade_cent ,
        a ~ dnorm( 0.5 , 0.25 ) ,
        bw ~ dlnorm( 0 , 0.25 ) ,
        bs ~ dlnorm( 0 , 0.25 ) ,
        bws ~ dlnorm( 0 , 0.25 ) ,
        sigma ~ dexp( 1 )
    ) , data=d )
```

Before looking at the fit, let's do prior simulation to see if these priors make sense.

R code 8.2

```
p <- extract.prior( m8M4a )
par(mfrow=c(1,3),cex=1.1) # 3 plots in 1 row
for ( s in -1:1 ) {
```

```
    idx <- which( d$shade_cent==s )
    plot( d$water_cent[idx] , d$blooms_std[idx] , xlim=c(-1,1) , ylim=c(0,1) ,
        xlab="water" , ylab="blooms" , pch=16 , col=rangi2 )
    mtext( concat( "shade = " , s ) )
    mu <- link( m8M4 , post=p , data=data.frame( shade_cent=s , water_cent=-1:1 ) )
    for ( i in 1:20 ) lines( -1:1 , mu[i,] , col=col.alpha("black",0.3) )
}
```

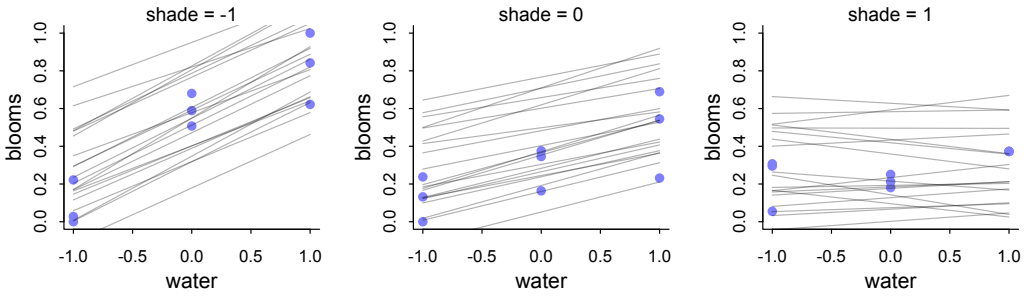


Wow, that's terrible. First guess was no good. We don't want to fit the data visually here, but we do want prior associations within the realm of possibility. So we need to make the prior means closer to zero. Let's try:

R code 8.3

```
m8M4b <- quap(
    alist(
        blooms_std ~ dnorm( mu , sigma ) ,
        mu <- a + bw*water_cent - bs*shade_cent - bws*water_cent*shade_cent ,
        a ~ dnorm( 0.5 , 0.25 ) ,
        bw ~ dlnorm( -2 , 0.25 ) ,
        bs ~ dlnorm( -2 , 0.25 ) ,
        bws ~ dlnorm( -2 , 0.25 ) ,
        sigma ~ dexp( 1 )
    ) , data=d )
```

Extract the prior and repeat the plotting code:



That's a lot better, mostly within the empirically possible range of the outcome at least. There is still plenty of uncertainty for the data to hone down, however.

8H1. Let's begin by loading the data and inspecting the `bed` variable.

R code 8.4

```
library(rethinking)
data(tulips)
d <- tulips
d$bed
```

```
 [1] a a a a a a a a a b b b b b b b b b c c c c c c c c c c
Levels: a b c
```

This is a factor with three levels. So we could either code two dummy variables to contain the same information or rather one index variable. Both approaches were introduced in Chapter 5. I'll show both approaches here, to provide additional examples.

First, the dummy variable approach. We'll need two dummy variables, one less than the number of categories. This will construct them:

R code 8.5

```
d$bed_b <- ifelse( d$bed=="b" , 1 , 0 )
d$bed_c <- ifelse( d$bed=="c" , 1 , 0 )
```

For bed `a`, it will be absorbed into the intercept, and then the coefficients for each dummy variable will contain *contrasts* with bed `a`.

And here is the model using these dummy variables. I'm also going to center the variables, just like in the chapter.

R code 8.6

```
d$blooms_std <- d$blooms / max(d$blooms)
d$water_cent <- d$water - mean(d$water)
d$shade_cent <- d$shade - mean(d$shade)
m1 <- quap(
    alist(
        blooms_std ~ dnorm( mu , sigma ),
        mu <- a + bw*water_cent + bs*shade_cent +
              bws*water_cent*shade_cent +
              b_bed_b*bed_b + b_bed_c*bed_c ,
        a ~ dnorm( 0 , 0.25 ),
        c(bw,bs,bws,b_bed_b,b_bed_c) ~ dnorm(0,0.25),
        sigma ~ dexp( 1 )
    ) , data=d )
precis(m1)
```

```
          mean   sd  5.5% 94.5%
a         0.27 0.03  0.21  0.32
bw        0.21 0.03  0.17  0.25
bs       -0.11 0.03 -0.15 -0.07
bws      -0.14 0.03 -0.19 -0.09
b_bed_b   0.12 0.05  0.04  0.20
b_bed_c   0.14 0.05  0.06  0.22
sigma     0.11 0.01  0.08  0.13
```

This table is a bit monstrous to look at, but comparing to the interaction model fit in the chapter reveals that everything is basically the same, except for the presence now of the bed parameters. Both beds "b" and "c" appear to have done better than bed "a" did. How can I tell? Because both `b_bed_b` and `b_bed_c` are reliably positive.

Now let's do the model over again, using an index variable instead. This approach estimates a unique intercept for each category. To construct the index variable, you can use the convenient `coerce_index` function:

R code 8.7
```
( d$bed_idx <- coerce_index( d$bed ) )
```

```
 [1] 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3
```

So now bed "a" has index 1, bed "b" index 2, and bed "c" index 3. To fit the model:

R code 8.8
```
m2 <- quap(
    alist(
        blooms_std ~ dnorm( mu , sigma ),
        mu <- a[bed_idx] + bw*water_cent + bs*shade_cent +
              bws*water_cent*shade_cent ,
        a[bed_idx] ~ dnorm( 0 , 0.25 ),
        c(bw,bs,bws) ~ dnorm( 0 , 0.25 ),
        sigma ~ dexp( 1 )
    ) , data=d )
precis(m2,depth=2)
```

```
       mean   sd  5.5% 94.5%
a[1]   0.26 0.04  0.21  0.32
a[2]   0.39 0.04  0.33  0.44
a[3]   0.40 0.04  0.34  0.46
bw     0.21 0.03  0.17  0.25
bs    -0.11 0.03 -0.15 -0.07
bws   -0.14 0.03 -0.19 -0.09
sigma  0.11 0.01  0.08  0.13
```

These are (nearly) the same estimates. Again we see that beds "b" and "c" did better than bed "a". Bed "c" did appear to grow a little better than bed "b". But what's the posterior distribution of that difference? We can calculate it with samples, just like the examples in Chapter 5:

R code 8.9
```
post <- extract.samples(m2)
diff_b_c <- post$a[,2] - post$a[,3]
PI( diff_b_c )
```

```
         5%         94%
-0.09631392  0.06708758
```

So while the expected difference is there, the posterior distribution of the difference has a lot of probability on both sides of zero.

So what to make of all of this? Including bed in the analysis doesn't change any qualitative inference about the experiment, even though there probably were differences between the beds.

8H2. Now let's compare the model from the previous problem with the interaction model from the chapter.

R code 8.10
```
m3 <- quap(
    alist(
        blooms_std ~ dnorm( mu , sigma ),
        mu <- a + bw*water_cent + bs*shade_cent +
              bws*water_cent*shade_cent ,
        a ~ dnorm( 0 , 0.25 ),
        c(bw,bs,bws) ~ dnorm( 0 , 0.25 ),
        sigma ~ dexp( 1 )
```

```
    ) , data=d )
compare(m2,m3)
```

```
    WAIC    SE dWAIC  dSE pWAIC weight
m2 -22.8 10.16   0.0   NA  10.0   0.57
m3 -22.3 10.18   0.6 8.35   6.4   0.43
```

The model with `bed`, `m2`, does a little bit better in the WAIC comparison. But the difference is very small. Why? Because while there is some evidence that bed mattered, the treatments mattered a lot more. So including `bed` in the model doesn't help prediction much, as least as far as WAIC can estimate. This is all as it should be: this was a factorial experiment. In the experimental design, there is no correlation between bed and treatment. So even when there are effects of bed, we can still get good measures of the treatments. In observational studies, trying to control for common confounds like bed is much more important, typically.

8H3. (a) Let's load the data and fit the model, as in the chapter:

R code 8.11

```
library(rethinking)
data(rugged)
d <- rugged
d$log_gdp <- log(d$rgdppc_2000)
dd <- d[ complete.cases(d$rgdppc_2000) , ]
dd$log_gdp_std <- dd$log_gdp / mean(dd$log_gdp)
dd$rugged_std <- dd$rugged / max(dd$rugged)
dd$cid <- ifelse( dd$cont_africa==1 , 1 , 2 )

m8.3 <- quap(
    alist(
        log_gdp_std ~ dnorm( mu , sigma ) ,
        mu <- a[cid] + b[cid]*( rugged_std - 0.215 ) ,
        a[cid] ~ dnorm( 1 , 0.1 ) ,
        b[cid] ~ dnorm( 0 , 0.3 ) ,
        sigma ~ dexp( 1 )
    ) , data=dd )
```

Now let's look at the Pareto *k* values, sorted and displayed with the country names:

R code 8.12

```
k <- PSISk( m8.3 )
o <- order( k , decreasing=TRUE )
data.frame( country=as.character( dd$isocode ) ,
    k=k , dd$rugged_std , dd$log_gdp_std )[o,]
```

```
    country     k dd.rugged_std dd.log_gdp_std
145     SYC  0.64  0.7876491454      1.1501264
150     TJK  0.49  0.8547242825      0.7826922
24      BWA  0.40  0.0291841342      1.0507430
118     NPL  0.35  0.8131247985      0.8438960
62      GNQ  0.34  0.0901322154      1.1304719
84      KGZ  0.34  0.6912286359      0.8632546
93      LSO  0.34  1.0000000000      0.8994088
27      CHE  0.28  0.7676555950      1.2110100
167     YEM  0.26  0.3745565946      0.7830325
```

```
55      GAB  0.25  0.0351499516      1.0237147
57      GEO  0.23  0.5899709771      0.8851899
108     MUS  0.22  0.1530151564      1.0768738
144     SWZ  0.22  0.4938729442      0.9922796
...
```

Note that the Pareto *k* values depend upon samples, so the values you see will be a little different. But the basic order should be similar. Seychelles (SYC) does have the largest Pareto *k* value. But several other countries also have pretty high values. What do these countries have in common? That are poorly fit the trend, but in different ways. Seychelles has an unusually high GDP. Tajikstan (TJK) has an unusually low GDP. Botswana (BWA) has an unusually high GDP for such a flat African country.

(b) Simply change the likelihood to Student-t:

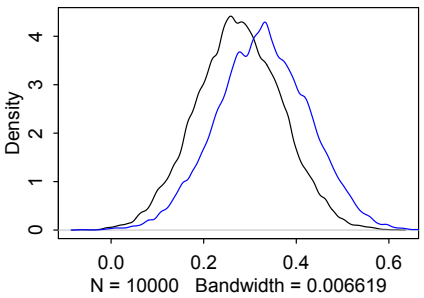
R code 8.13

```
m8.3t <- quap(
    alist(
        log_gdp_std ~ dstudent( 2 , mu , sigma ) ,
        mu <- a[cid] + b[cid]*( rugged_std - 0.215 ) ,
        a[cid] ~ dnorm( 1 , 0.1 ) ,
        b[cid] ~ dnorm( 0 , 0.3 ) ,
        sigma ~ dexp( 1 )
    ) , data=dd )
```

We want to know how the posterior distribution of the difference between `b[1]` and `b[2]` has changed by using the Student-t distribution. So let's do that comparison directly:

R code 8.14

```
postN <- extract.samples( m8.3 )
postT <- extract.samples( m8.3t )
diffN <- postN$b[,1] - postN$b[,2]
diffT <- postT$b[,1] - postT$b[,2]
dens( diffN )
dens( diffT , add=TRUE , col="blue" )
```



Blue is the Student-t model. The difference has actually grown a little. The thing about the Student-t model is that it treats all the of the extreme points, not just the ones you notice with your eyes. So it can sometimes surprise you.

8H4. (a) You could have chosen to fit a number of different models. I'm going to fit three models that represent a basic analysis of the hypothesis: (1) A model predicting log-lang-per-capita with only a constant, (2) a model with only mean growing season as a predictor, and (3) a model that includes log(area) as a covariate. Then I'll compare them, using WAIC.

But keep in mind that WAIC is not choosing a model for us. The causal theory chooses the model. In this case that means both area and growing season influence the number of languages. These two variables may be associated with one another though other processes. For example, as a consequence of world history, larger national territories (area) may be found in particular places and therefore be associated with particular growing seasons. Since larger nations (area) tend to have fewer languages, there are clearly historical processes at work here. Can you make a DAG that expresses this?

Let's load the data and build some models:

R code 8.15

```
library(rethinking)
data(nettle)
d <- nettle
d$L <- standardize( log( d$num.lang / d$k.pop ) )
d$A <- standardize( log(d$area) )
d$G <- standardize( d$mean.growing.season )

m0 <- quap(
    alist(
        L ~ dnorm(mu,sigma),
        mu <- a + 0,
        a ~ dnorm(0,0.2),
        sigma ~ dexp(1)
    ) , data=d )

m1 <- quap(
    alist(
        L ~ dnorm(mu,sigma),
        mu <- a + bG*G,
        a ~ dnorm(0,0.2),
        bG ~ dnorm(0,0.5),
        sigma ~ dexp(1)
    ) , data=d )

m2 <- quap(
    alist(
        L ~ dnorm(mu,sigma),
        mu <- a + bG*G + bA*A,
        a ~ dnorm(0,0.2),
        c(bG,bA) ~ dnorm(0,0.5),
        sigma ~ dexp(1)
    ) , data=d )

compare(m0,m1,m2)
```

```
    WAIC    SE dWAIC  dSE pWAIC weight
m2 205.6 16.10   0.0   NA   4.7   0.53
m1 206.0 15.57   0.3 3.90   3.7   0.46
m0 213.7 17.22   8.1 8.09   2.7   0.01
```

This is very strong support for using mean growing season as a predictor of language diversity. Controlling for log(area) has little effect, as can be seen by comparing the estimates from models `m1` and `m2`:

R code 8.16

```
coeftab(m1,m2)
```

```
      m1      m2
```

```
a           0       0
bG       0.34    0.29
sigma    0.92    0.91
bA         NA   -0.17
nobs       74      74
```

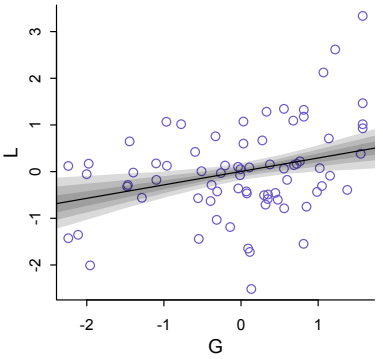
You can check the standard errors, too, to ensure that the estimate for `mean.growing.season` is reliably positive in both cases.

Plotting the predicted relationship for `m2`:

R code 8.17

```
G_seq <- seq( from=-2.5 , to=2 , length.out=30 )
new_dat <- data.frame( A=0 , G=G_seq )

mu <- link( m2 , data=new_dat )

plot( L ~ G , data=d , col="slateblue" )
lines( G_seq , colMeans(mu) )

# plot several intervals with shading
for ( p in c(0.5,0.79,0.95) ) {
mu_PI <- apply( mu , 2 , PI , prob=p )
shade( mu_PI , G_seq )
}
```

And here's the plot:



Hm, that relationship doesn't actually look very linear. Most of the error is above the line, at long growing season values on the right side. In any event, there does seem to be some positive relationship between the two variables, as predicted. Maybe we shouldn't be surprised by how poor the linear relationship represents the data. After all, there are a ton of unmeasured variables that influence language density. For example, imperial expansions have mainly reduced language diversity, but have done so unequally in different regions.

(b) Fitting analogous models and comparing them (`m0` is same as before):

R code 8.18

```
d$S <- standardize( d$sd.growing.season )
m4 <- quap(
    alist(
        L ~ dnorm(mu,sigma),
        mu <- a + bS*S,
        a ~ dnorm(0,0.2),
```

```
        bS ~ dnorm(0,0.5),
        sigma ~ dexp(1)
    ) , data=d )
m5 <- quap(
    alist(
        L ~ dnorm(mu,sigma),
        mu <- a + bS*S + bA*A,
        a ~ dnorm(0,0.2),
        c(bS,bA) ~ dnorm(0,0.5),
        sigma ~ dexp(1)
    ) , data=d )
compare(m0,m4,m5)
```

```
    WAIC    SE dWAIC  dSE pWAIC weight
m4 211.0 17.36   0.0   NA   3.7   0.51
m5 211.8 17.21   0.8 3.86   5.4   0.34
m0 213.5 17.11   2.6 4.37   2.5   0.14
```

A very similar story, although not as strong a relationship. Take a look at the marginal posterior from `m5`:

```
precis(m5)
```

R code 8.19

```
       mean   sd  5.5% 94.5%
a      0.00 0.10 -0.15  0.15
bS    -0.14 0.12 -0.34  0.06
bA    -0.19 0.12 -0.39  0.01
sigma  0.94 0.08  0.82  1.06
```

Including log(area) generates considerable uncertainty about the direction of the effect of `sd.growing.season`. Still, the MAP is negative, as predicted. Of course, there is no reason to think the effect of S should be linear, as this model assumes.

(c) Here's the interaction model:

```
m6 <- quap(
    alist(
        L ~ dnorm(mu,sigma),
        mu <- a + bG*G + bS*S + bGS*G*S,
        a ~ dnorm(0,0.2),
        c(bG,bS,bGS) ~ dnorm(0,0.5),
        sigma ~ dexp(1)
    ) , data=d )
```

R code 8.20

Okay, so let's inspect the marginal posterior:

```
precis(m6)
```

R code 8.21

```
       mean   sd  5.5% 94.5%
a      0.00 0.09 -0.14  0.15
bG     0.23 0.11  0.05  0.41
bS    -0.23 0.10 -0.39 -0.07
bGS   -0.24 0.10 -0.40 -0.08
sigma  0.85 0.07  0.74  0.97
```

You can see above that the interaction coefficient is reliably negative. What that means exactly will require some more work.

Let's see what the predictions look like. Let's show the interaction in a panel of six plots. The top row will show the relationship between log-lang-per-capita and mean.growing.season, across values of sd.growing.season. The bottom row will show log-lang-per-capita on sd.growing.season, across values of mean.growing.season. This is just to show the two-way interaction from both perspectives.

I'm also going to use transparency, as a function of distance from the value on the top of each plot, to show how to data change through the 3rd, un-plotted, dimension. The function `col.dist` in the `rethinking` package handles this for you. I didn't use it in the book, so here is an example of how it can be helpful. You can get more details about how it works from the help `?col.dist`. The key issue is the standard deviation value, which determines how quickly color fades as individual points move away from some reference value. The reference value in these plots will be the value of the third variable displayed on the top margin. You'll want to play with the standard deviation value to get a sense of how it works. Larger values mean less fading.

This code will draw a triptych of interactions, varying `mean.growing.season` along the horizontal axis and `sd.growing.season` across the plots.

R code 8.22

```
# pull out 10%, 50%, and 95% quantiles of sd.growing.season
# these values will be used to make the three plots
S_seq <- quantile(d$S,c(0.1,0.5,0.95))

# now loop over the three plots
# draw languages against mean.growing.season in each
G_seq <- seq(from=-2.5,to=2,length.out=30)
par(mfrow=c(1,3),cex=1.1) # set up plot window for row of 3 plots
for ( i in 1:3 ) {
    S_val <- S_seq[i] # select out value for this plot
    new.dat <- data.frame(
        G = G_seq,
        S = S_val )
    mu <- link( m6 , data=new.dat )
    mu.mean <- apply( mu , 2 , mean )
    mu.PI <- apply( mu , 2 , PI )

    # fade point color as function of distance from sd.val
    cols <- col.dist( d$S , S_val , 2 , "slateblue" )

    plot( L ~ G , data=d , col=cols , lwd=2 )
    mtext( paste("S =",round(S_val,2)) , 3 )
    lines( G_seq , mu.mean )
    shade( mu.PI , G_seq )
}
```

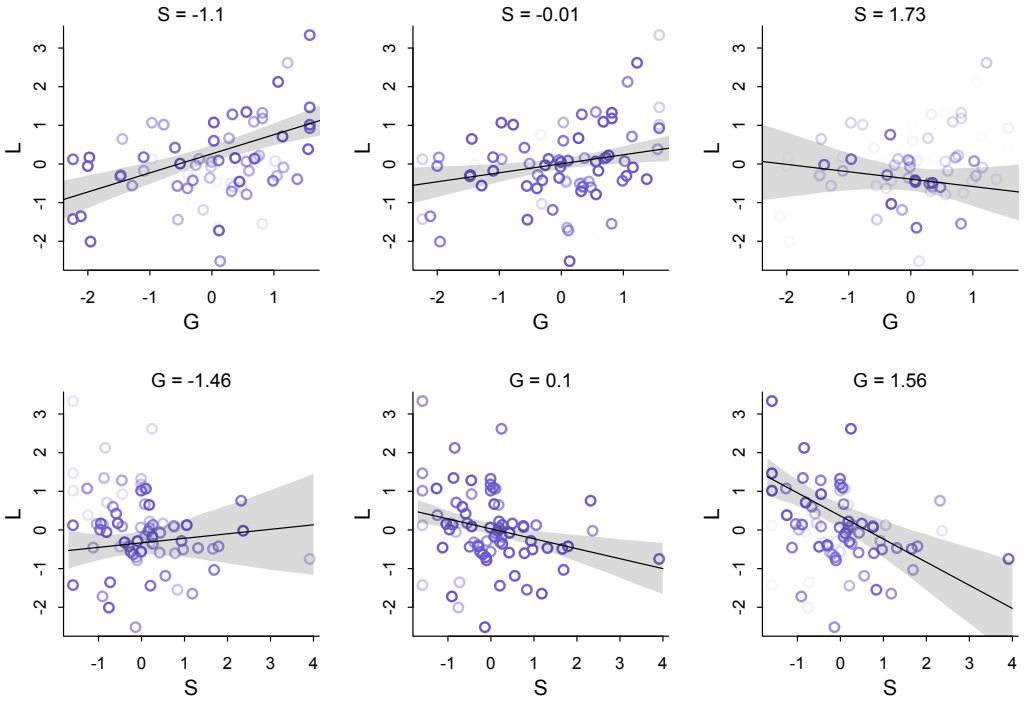
And this code will produce the analogous triptych in which `sd.growing.season` is varied on the horizontal axis.

R code 8.23

```
# pull out 10%, 50%, and 95% quantiles of mean.growing.season
G_seq <- quantile(d$G,c(0.1,0.5,0.95))

# now loop over the three plots
x.seq <- seq(from=-1.7,to=4,length.out=30)
par(mfrow=c(1,3),cex=1.1) # set up plot window for row of 3 plots
for ( i in 1:3 ) {
```

```
    G_val <- G_seq[i] # select out value for this plot
    new.dat <- data.frame(
        G = G_val ,
        S = x.seq )
    mu <- link( m6 , data=new.dat )
    mu.mean <- apply( mu , 2 , mean )
    mu.PI <- apply( mu , 2 , PI )

    # fade point color as function of distance from sd.val
    cols <- col.dist( d$G , G_val , 5 , "slateblue" )

    plot( L ~ S , data=d , col=cols , lwd=2 )
    mtext( paste("G =",round(G_val,2)) , 3 )
    lines( x.seq , mu.mean )
    shade( mu.PI , x.seq )
}
```

And here are both triptych constructions:



So the models suggest that mean growing season increases language diversity, unless the variance in growing season is also high (top row). Simultaneously, variance in growing season decreases language diversity, unless the mean growing season is very short (bottom row).

8H5. Load the data and construct the variables we'll need:

R code 8.24

```
library(rethinking)
data(Wines2012)
```

```
d <- Wines2012

dat_list <- list(
    S = standardize(d$score),
    jid = as.integer(d$judge),
    wid = as.integer(d$wine)
)
```

The model is straightforward. The only issue is the priors. Since I've standardized the outcome, we can use the ordinary N(0,0.5) prior from the examples in the text with standardized outcomes. Then the prior outcomes will stay largely within the possible outcome space. A bit more regularization than that wouldn't be a bad idea either.

R code 8.25

```
m1 <- quap(
    alist(
        S ~ dnorm( mu , sigma ),
        mu <- a[jid] + w[wid],
        a[jid] ~ dnorm(0,0.5),
        w[wid] ~ dnorm(0,0.5),
        sigma ~ dexp(1)
    ), data=dat_list )
```

Since this is your first MCMC homework, we'll spend some time inspecting the chains to ensure they worked. First, the diagnostics that `precis` provides:

R code 8.26

```
precis( m1 , 2 )
```

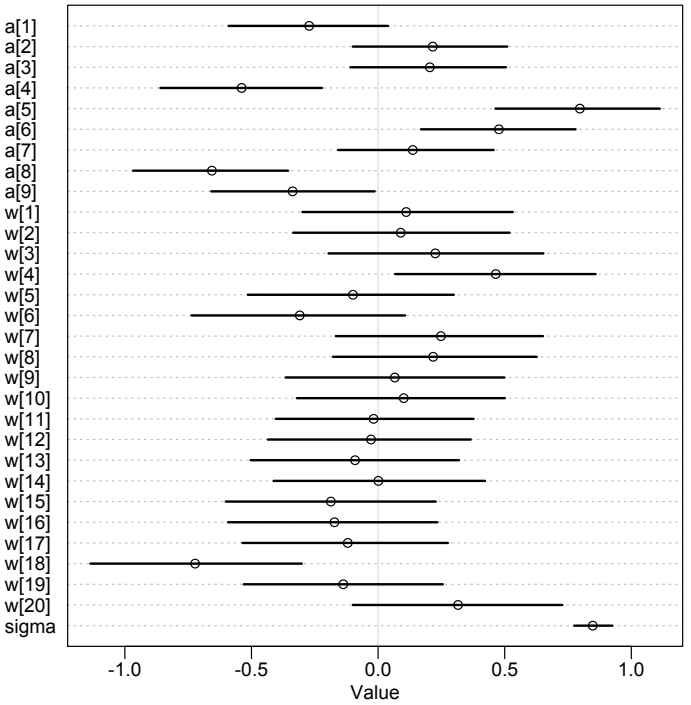
```
       mean   sd  5.5% 94.5%
a[1]  -0.28 0.19 -0.58  0.01
a[2]   0.22 0.19 -0.08  0.51
a[3]   0.21 0.19 -0.09  0.51
a[4]  -0.55 0.19 -0.85 -0.25
a[5]   0.81 0.19  0.51  1.11
a[6]   0.48 0.19  0.19  0.78
a[7]   0.13 0.19 -0.16  0.43
a[8]  -0.67 0.19 -0.97 -0.37
a[9]  -0.35 0.19 -0.65 -0.05
w[1]   0.12 0.25 -0.27  0.51
w[2]   0.09 0.25 -0.30  0.48
w[3]   0.24 0.25 -0.16  0.63
w[4]   0.48 0.25  0.09  0.87
w[5]  -0.11 0.25 -0.50  0.28
w[6]  -0.32 0.25 -0.71  0.07
w[7]   0.25 0.25 -0.14  0.64
w[8]   0.24 0.25 -0.16  0.63
w[9]   0.07 0.25 -0.32  0.46
w[10]  0.10 0.25 -0.29  0.50
w[11] -0.01 0.25 -0.40  0.38
w[12] -0.03 0.25 -0.42  0.37
w[13] -0.09 0.25 -0.48  0.30
w[14]  0.01 0.25 -0.39  0.40
w[15] -0.19 0.25 -0.58  0.20
w[16] -0.17 0.25 -0.57  0.22
```

```
w[17] -0.12 0.25 -0.52  0.27
w[18] -0.75 0.25 -1.14 -0.35
w[19] -0.14 0.25 -0.53  0.25
w[20]  0.33 0.25 -0.06  0.73
sigma  0.79 0.04  0.72  0.85
```

Now let's plot these parameters so they are easier to interpret:

R code 8.27

```
plot( precis( m1 , 2 ) )
```



The a parameters are the judges. Each represents an average deviation of the scores. So judges with lower values are harsher on average. Judges with higher values liked the wines more on average. There is some noticeable variation here. It is fairly easy to tell the judges apart.

The w parameters are the wines. Each represents an average score across all judges. Except for wine 18 (a New Jersey red I think), there isn't that much variation. These are good wines, after all. Overall, there is more variation from judge than from wine.

8H6. The easiest way to code the data is to use indicator variables. Let's look at that approach first. I'll do an index variable version next. I'll use the three indicator variables W (NJ wine), J (American NJ), and R (red wine).

R code 8.28

```
dat_list2 <- list(
    S = standardize(d$score),
    W = d$wine.amer,
    J = d$judge.amer,
    R = ifelse(d$flight=="red",1L,0L)
)
```

The model structure is just a linear model with an ordinary intercept. I'll put a relatively tight prior on the intercept, since it must be near zero (centered outcome). What about the coefficients for the indicator variables? Let's pretend we haven't already seen the results from Problem 1—there aren't any big wine differences to find there. Without that cheating foresight, we should consider what the most extreme effect could be. How big could the difference between NJ and French wines be? Could it be a full standard deviation? If so, then maybe a Normal(0,0.5) prior makes sense, since they place a full standard deviation difference out in the tails of the prior. I'd personally be inclined to something even tighter, so that it regularizes more. But let's go with these wide priors, which nevertheless stay within the outcome space. It would make even more sense to put a tighter prior on the difference between red and white wines—on average they should be the no different, because judges only compare within flights. Here's the model:

R code 8.29

```
m2a <- quap(
    alist(
        S ~ dnorm( mu , sigma ),
        mu <- a + bW*W + bJ*J + bR*R,
        a ~ dnorm( 0 , 0.2 ),
        c(bW,bJ,bR) ~ dnorm( 0 , 0.5 ),
        sigma ~ dexp(1)
    ), data=dat_list2 )
precis( m2a )
```

```
       mean   sd  5.5% 94.5%
a     -0.01 0.12 -0.21  0.18
bW    -0.18 0.13 -0.40  0.04
bJ     0.23 0.13  0.01  0.44
bR    -0.01 0.14 -0.22  0.21
sigma  0.98 0.05  0.90  1.06
```

As expected, red and wines are on average the same—`bR` is right on top of zero. American judges seem to be more on average slightly more generous with ratings—`bJ` is slightly but reliably above zero. American wines have slightly lower average ratings than French wines—`bW` is mostly below zero, but not very large in absolute size.

Okay, now for an index variable version. The thing about index variables is that you can easily end up with more parameters than in an equivalent indicator variable model. But it's still the same posterior distribution. You can convert from one to the other (if the priors are also equivalent). We'll need three index variables:

R code 8.30

```
dat_list2b <- list(
    S = standardize(d$score),
    wid = d$wine.amer + 1L,
    jid = d$judge.amer + 1L,
    fid = ifelse(d$flight=="red",1L,2L)
)
```

Now `wid` is 1 for a French wine and 2 for a NJ wine, `jid` is 1 for a French judge and 2 for an American judge, and `fid` is 1 for red and 2 for white. Those `1L` numbers are just the R way to type the number as an integer—"`1L`" is the integer 1, while "`1`" is the real number 1. We want integers for an index variable.

Now let's think about priors for the parameters that correspond to each index value. Now the question isn't how big the difference could be, but rather how far from the mean an indexed category could be. If we use Normal(0,0.5) priors, that would make a full standard deviation difference from the global mean rare. It will also match what we had above, in a crude sense. Again, I'd be tempted

to something narrow, for the sake of regularization. But certainly something like Normal(0,10) is flat out silly, because it makes impossible values routine. Let's see what we get:

R code 8.31

```
m2b <- quap(
    alist(
        S ~ dnorm( mu , sigma ),
        mu <- w[wid] + j[jid] + f[fid],
        w[wid] ~ dnorm( 0 , 0.5 ),
        j[wid] ~ dnorm( 0 , 0.5 ),
        f[wid] ~ dnorm( 0 , 0.5 ),
        sigma ~ dexp(1)
    ), data=dat_list2b )
precis( m2b , 2 )
```

```
       mean   sd  5.5% 94.5%
w[1]   0.09 0.30 -0.39  0.57
w[2]  -0.09 0.30 -0.57  0.39
j[1]  -0.12 0.30 -0.60  0.36
j[2]   0.12 0.30 -0.36  0.60
f[1]   0.00 0.30 -0.48  0.48
f[2]   0.00 0.30 -0.47  0.48
sigma  0.98 0.05  0.90  1.06
```

To see that this model is the same as the previous, let's compute contrasts. The contrast between American and French wines is:

R code 8.32

```
post <- extract.samples(m2b)
diff_w <- post$w[,2] - post$w[,1]
precis( diff_w )
```

```
'data.frame': 10000 obs. of 1 variables:
        mean   sd  5.5% 94.5%    histogram
diff_w -0.18 0.15 -0.42  0.05 ▁▁▂▅▇▂▁▁
```

That's almost exactly the same mean and standard deviation as `bW` in the first model. The other contrasts match as well.

8H7. I'll use the indicator variable approach here, because it'll be much easier. Once you start using MCMC in the next chapter, it'll be possible to define very flexible parameter structures. Then the index approach will be easy again.

For the indicator approach, we can use the same predictor variables as before:

R code 8.33

```
dat_list2 <- list(
    S = standardize(d$score),
    W = d$wine.amer,
    J = d$judge.amer,
    R = ifelse(d$flight=="red",1L,0L)
)
```

It's the model that is different.

R code 8.34

```
m3 <- quap(
    alist(
```

```
        S ~ dnorm( mu , sigma ),
        mu <- a + bW*W + bJ*J + bR*R +
              bWJ*W*J + bWR*W*R + bJR*J*R,
        a ~ dnorm(0,0.2),
        c(bW,bJ,bR) ~ dnorm(0,0.5),
        c(bWJ,bWR,bJR) ~ dnorm(0,0.25),
        sigma ~ dexp(1)
    ), data=dat_list2 )
```

I used the same priors as before for the main effects. I used tighter priors for the interactions. Why? Because interactions represent sub-categories of data, and if we keep slicing up the sample, differences can't keep getting bigger. Again, the most important thing is not to use flat priors like Normal(0,10) that produce impossible outcomes.

R code 8.35

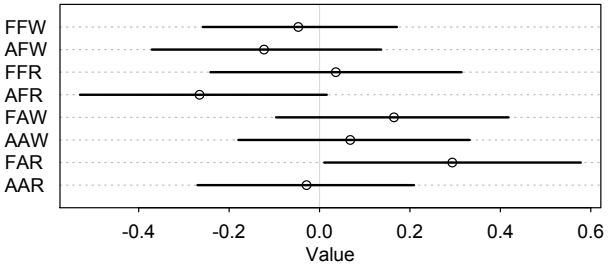
```
precis(m3)
```

```
       mean   sd  5.5% 94.5%
a     -0.05 0.13 -0.25  0.16
bW    -0.07 0.17 -0.35  0.20
bJ     0.21 0.18 -0.08  0.49
bR     0.09 0.18 -0.21  0.38
bWJ   -0.02 0.18 -0.31  0.27
bWR   -0.23 0.18 -0.52  0.06
bJR    0.05 0.18 -0.24  0.34
sigma  0.98 0.05  0.89  1.06
```

Reading the parameters this way is not easy. But right away you might notice that `bW` is now close to zero and overlaps it a lot on both sides. NJ wines are no longer on average worse. So the interactions did something. Glancing at the interaction parameters, you can see that only one of them has much mass away from zero, `bWR`, the interaction between NJ wines and red flight, so red NJ wines. To get the predicted scores for red and white wines from both NJ and France, for both types of judges, we can use `link`:

R code 8.36

```
pred_dat <- data.frame(
    W = rep( 0:1 , times=4 ),
    J = rep( 0:1 , each=4 ),
    R = rep( c(0,0,1,1) , times=2 )
)
mu <- link( m3 , data=pred_dat )
row_labels <- paste( ifelse(pred_dat$W==1,"A","F") ,
                 ifelse(pred_dat$J==1,"A","F") ,
                 ifelse(pred_dat$R==1,"R","W") , sep="" )
plot( precis( list(mu=mu) , 2 ) , labels=row_labels )
```



I've added informative labels. FFW means: French wine, French judge, White wine. So the first four rows are as judged by French judges. The last four are as judged by American judges. The two rows that jump out are the 4th and the 2nd-to-last, AFR and FAR. Those are NJ red wines as judged by French judges and French red wines as judged by American judges. French judges didn't like NJ reds so much (really only one NJ red, if you look back at Problem 1). And American judges liked French reds more. Besides these two interactions, notice that it is very hard to figure this out from the table of coefficients.