chapter4_solutions

CHAPTER 4 PRACTICE PROBLEMS

Easy.

4E1. In the model definition below, which line is the likelihood? yi = Normal(mu, sigma) mu = Normal(0, 10) sigma = Exponential(1)

```
library(rethinking)
```

```
## Loading required package: rstan
```

```
## Loading required package: StanHeaders
```

```
## Loading required package: ggplot2
```

```
## rstan (Version 2.21.2, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
## Loading required package: parallel
```

```
## rethinking (Version 2.13)
```

```
##
## Attaching package: 'rethinking'
```

```
## The following object is masked from 'package:stats':
##
##     rstudent
```

```
# yi = Normal(mu, sigma) (the other two are priors)
# a is saying, what is likelihood of data (yi) given a distribution
# (Normal(mu, sigma))
# the other 2 are saying, "what is the probability of parameter mu
# or sigma, given a fixed distribution?"
```

4E2. In the model definition just above, how many parameters are in the posterior distribution?

```
# 2 : the posterior distribution would have mu and sigma
```

4E3. Using the model definition above, write down the appropriate form of Bayes' theorem that includes the proper likelihood and priors.

```
# ?
# had to looks this one up
# P(mu, sigma | y_i) = Likelihood * Prior
#                    = L(y | mu, sigma) * P(mu) * P(sigma)
```

4E4. In the model definition below, which line is the linear model? yi = Normal(mu, sigma) mui = alpha + beta*xi alpha = Normal(0, 10) beta = Normal(0, 1) sigma = Exponential(2)

```
# mu_i = alpha + beta*x_i
# the first is likelihood, last 3 are priors for alpha, beta, and sigma
```

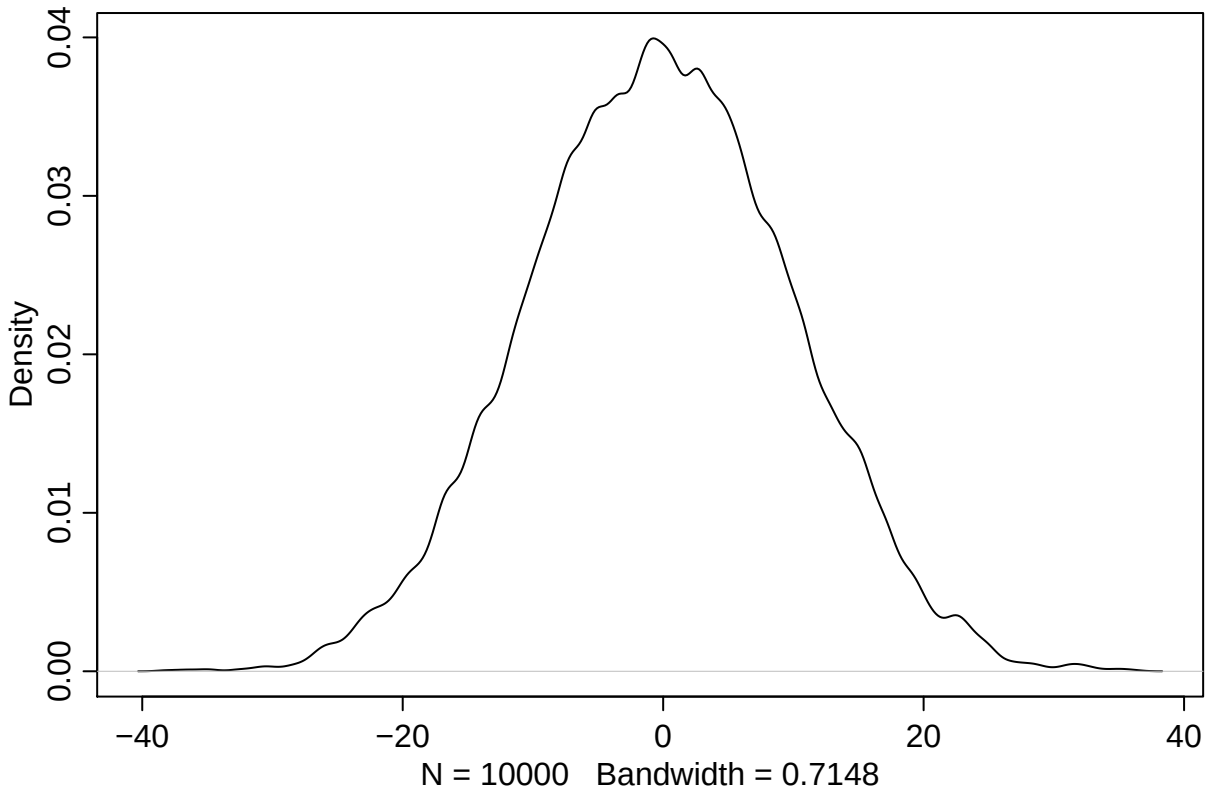
4E5. In the model definition just above, how many parameters are in the posterior distribution?

```
# 3: alpha, beta, sigma
```

Medium.

4M1. For the model definition below, simulate observed y values from the prior (not the posterior). yi = Normal(mu, sigma) mu = Normal(0, 10) sigma = Exponential(1)

```
sample_mu <- rnorm(1e4, 0, 10)
sample_sigma <- rexp(1e4, rate = 1)

prior_y <- rnorm(1e4, sample_mu, sample_sigma)
dens(prior_y)
```



4M2. Translate the model just above into a quap formula.

```
# translate model into quap formula
flist <- alist(y ~ dnorm(mu, sigma),
               mu ~ dnorm(0, 10),
               sigma ~ dexp(rate = 1))
```

4M3. Translate the quap model formula below into a mathematical model definition.

```
# proposed quap model
flist <- alist(
y ~ dnorm( mu , sigma ),
```

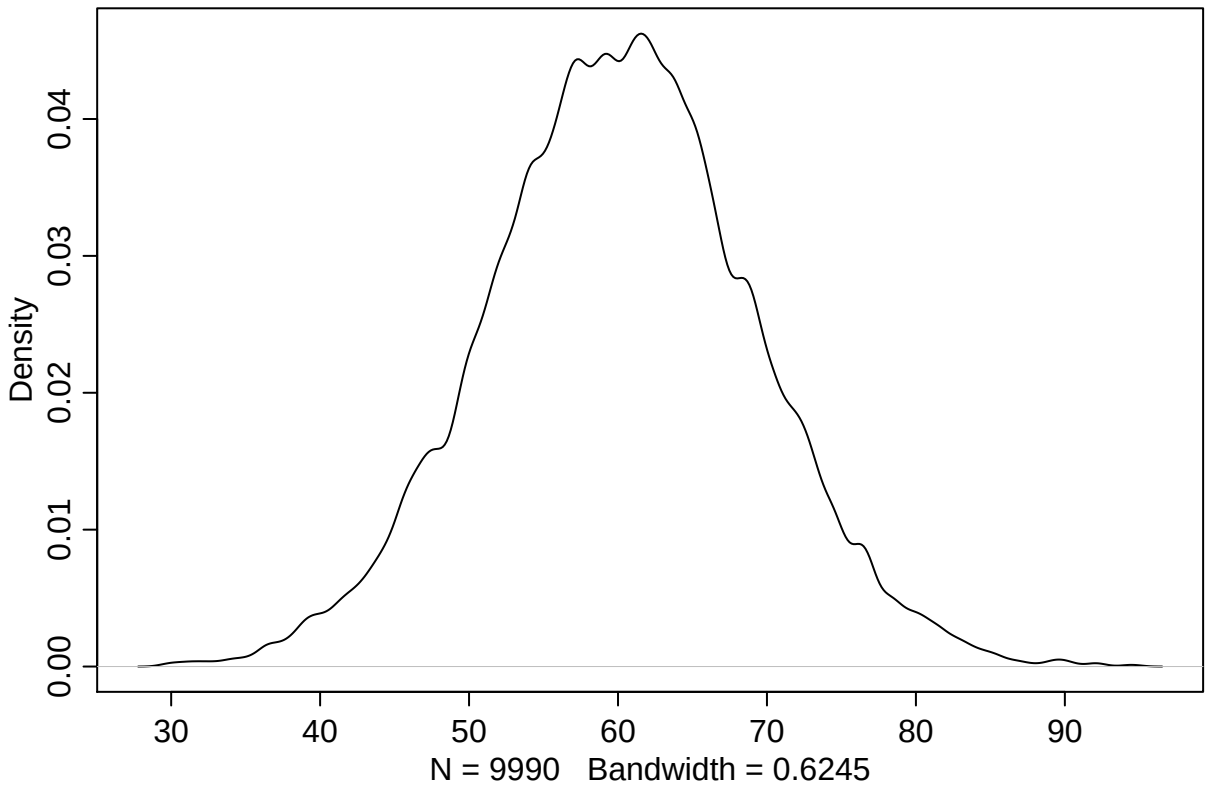
```
mu <- a + b*x,
a ~ dnorm( 0 , 10 ),
b ~ dunif( 0 , 1 ),
sigma ~ dexp( 1 )
)
# mathematical transformation
#y_i = Normal(mu, sigma)
#mu = alpha + beta* x
#alpha = Normal(0, 10)
#beta = Uniform(0, 1)
#sigma = Exponential(1)
```

4M4. A sample of students is measured for height each year for 3 years. After the third year, you want to fit a linear regression predicting height using year as a predictor. Write down the mathematical model definition for this regression, using any variable names and priors you choose. Be prepared to defend your choice of priors.

```
# average height = 60 inches
# sd = 6 inches

#h_i = Normal(mu, sigma)
#u = Normal(60, 6)
#sigma = Normal(6, 2)

# PRIOR PREDICTIVE CHECK
N <- 1000
mus <- rnorm(1000, 60, 6)
sigmas <- rnorm(1000, 6, 2)

prior_heights <- rnorm(1e4, mus, sigmas)
```

```
## Warning in rnorm(10000, mus, sigmas): NAs produced
```

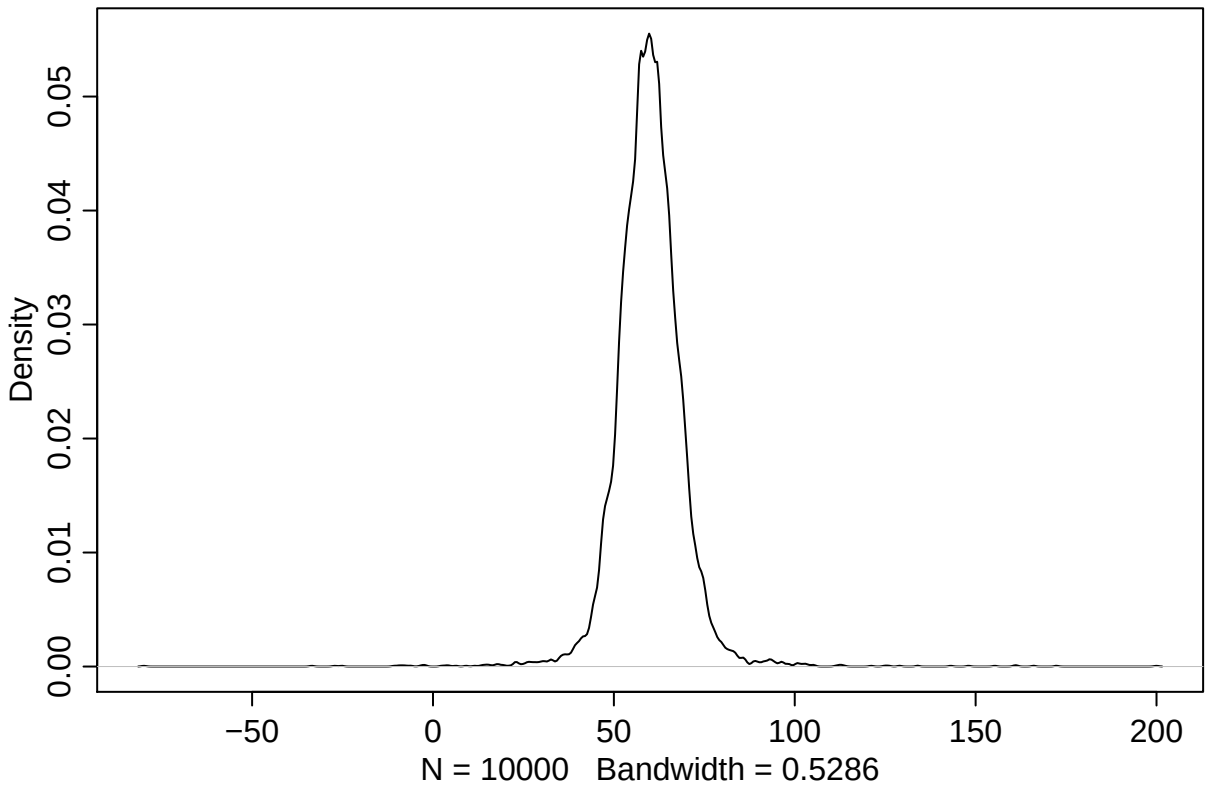
```
dens(prior_heights)
```



The model mostly thinks students are 60 inches tall, and most students are between 40 inches (3.5 fet) and 60 inches (6.5 feet), a reasonable human range for students.

4M5. Now suppose I remind you that every student got taller each year. Does this information lead you to change your choice of priors? How?

```
# probably not that much? average growth per year is probably small relative to mean height anyway so i


# better answer:
#sigma = Lognormal(1, 1)

N <- 1000
mus <- rnorm(1000, 60, 6)
sigmas <- rlnorm(1000, 1, 1)

prior_heights <- rnorm(1e4, mus, sigmas)
dens(prior_heights)
```

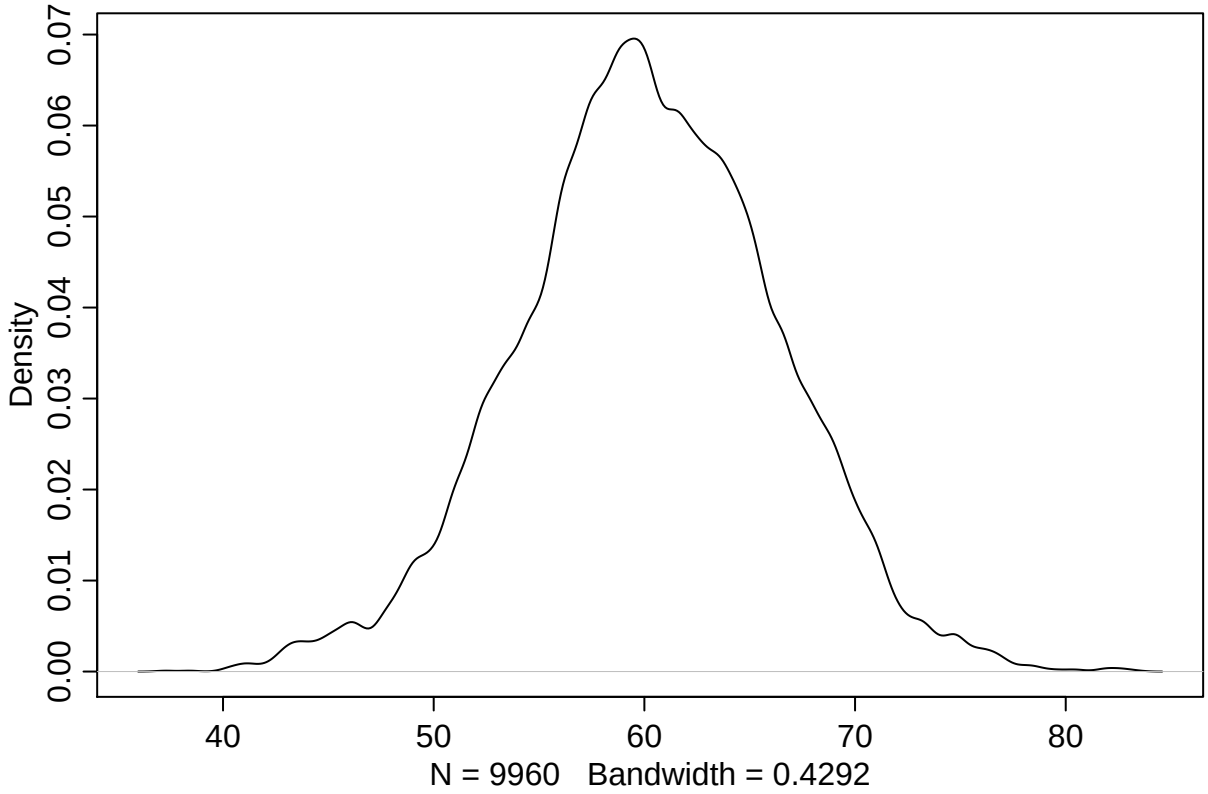


4M6. Now suppose I tell you that the variance among heights for students of the same age is never more than 64cm. How does this lead you to revise your priors?

```
# this implies sd is never more than 8 cm or 3 inches

# i'd update my prior for sigma:
#h_i = Normal(mu, sigma)
#u = Normal(60, 6)
#sigma = Normal(1.5, .5) # 3 standard deviations away from mean would be 3

# better answer:
#sigma = Lognormal(1, 1)

# PRIOR PREDICTIVE CHECK
N <- 1000
mus <- rnorm(1000, 60, 6)
sigmas <- rnorm(1000, 1.5, .5)

prior_heights <- rnorm(1e4, mus, sigmas)
```

```
## Warning in rnorm(10000, mus, sigmas): NAs produced
```

```
dens(prior_heights)
```



#

Hard. **4H1. The weights listed below were recorded in the !Kung census, but heights were not recorded for these individuals. Provide predicted heights and 89% intervals for each of these individuals.**

To solve this, we need the following four steps:

- Build a model to predict height, taking weight into account
- Use the model to simulate possible heights using the weights provided
- Compute the mean (median?) height of each simulation for the point estimate (predicted height) for each weight value
- Compute the 89% PI for each weight

```
# weights with unknown heights
unknown_weights = c(32.59, 46.95, 43.72,  32.59, 54.63, 64.78)
```

Step 1: Build Model

```
# load !Kung census data
library(rethinking)
data(Howell1)
d <- Howell1

# WE"LL USE A NORMAL MODEL TO GENERAGE PREDICTIONS

# assume height is normally distd with mean mu and sd sigma
h ~ Normal(mu, sigma)
```

```
## h ~ Normal(mu, sigma)
```

```
# assume height is a linear function of weight
u ~ alpha + beta(w - wbar) # w = weight, wbar = mean weight
```

```
## u ~ alpha + beta(w - wbar)
```

```
# when weight is average (135), height is around 150
# with not much variation
alpha ~ Normal(150, 5)
```

```
## alpha ~ Normal(150, 5)
```

```
# not really sure why we would use

# assume that height must not decrease as weight increases
beta ~ Uniform(0, 5)
```

```
## beta ~ Uniform(0, 5)
```

```
# standard deviations can't be negative
sigma ~ LogNormal(30,10) # could also use log-normal
```

```
## sigma ~ LogNormal(30, 10)
```

```
# why not a log-normal prior here?

# define average weight, wbar
wbar <- mean(d$weight)

# USE QUAP TO MAKE THE POSTERIOR
flist = alist(
height ~ dnorm( mu , sigma ),
mu <- a + b*(weight - wbar),
a ~ dnorm( 150 , 5 ),
#b = dunif( 0 , 5 ),
#sigma = dlnorm( 30, 10 ) (annonying and confusing why it won't work with these priors for b and sigma.
b ~ dlnorm(0,1),
sigma ~ dunif(0, 50)
)

model = quap(flist, data = d)
```

Step 2: Simulate heights using the model

The `sim` function will extract samples from the posterior and simulate heights using the samples.

```
# weight values of interest
unknown_weights = c(32.59, 46.95, 43.72,  32.59, 54.63, 64.78)
simulated_heights <- sim(model, data = list(weight = unknown_weights))
```

Step 3: Calculate the 89% PI of height and mean height for each weight provided

```
# calculate 89% interval for each weight
unknown_weights = c(32.59, 46.95, 43.72,  32.59, 54.63, 64.78)
height.PI <- apply( simulated_heights , 2 , PI , prob=0.89 )
height.PI
```

```
##         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
## 5%  117.8908 144.9926 137.3497 117.2343 156.8164 175.0377
## 94% 147.7459 172.5361 167.7556 147.3760 185.9594 204.7111
```

```
height.mean <- apply( simulated_heights , 2 , mean )
height.mean
```

```
## [1] 132.9557 158.3972 152.7992 133.0605 171.4741 189.8031
```

```
# individual weighing 32 kg is between 118 and 148 cm tall
# individual weighting 64 kg is between 175 and 205 cm tall
```

Step 4: Neaten up the output so it looks good

```
library(data.table)
individual = c(1,2,3,4,5,6)
unknown_weights = c(32.59, 46.95, 43.72,  32.59, 54.63, 64.78)
individual_mean_height = apply( simulated_heights , 2 , mean )
individual_interval = apply( simulated_heights , 2 , PI , prob=0.89 )

output = data.table(individual = c(1,2,3,4,5,6),
                    weight = unknown_weights,
                    lower_bound = individual_interval[1,],
                    expected_height = apply( simulated_heights , 2 , mean ),
                    upper_bound = individual_interval[2,])
output
```

```
##    individual weight lower_bound expected_height upper_bound
## 1:          1  32.59    117.8908        132.9557    147.7459
## 2:          2  46.95    144.9926        158.3972    172.5361
## 3:          3  43.72    137.3497        152.7992    167.7556
## 4:          4  32.59    117.2343        133.0605    147.3760
## 5:          5  54.63    156.8164        171.4741    185.9594
## 6:          6  64.78    175.0377        189.8031    204.7111
```

4H2. Select out all the rows in the Howell1 data with ages below 18 years of age. If you do it right, you should end up with a new data frame with 192 rows in it.

```
d2 = d[d$age < 18,]
```

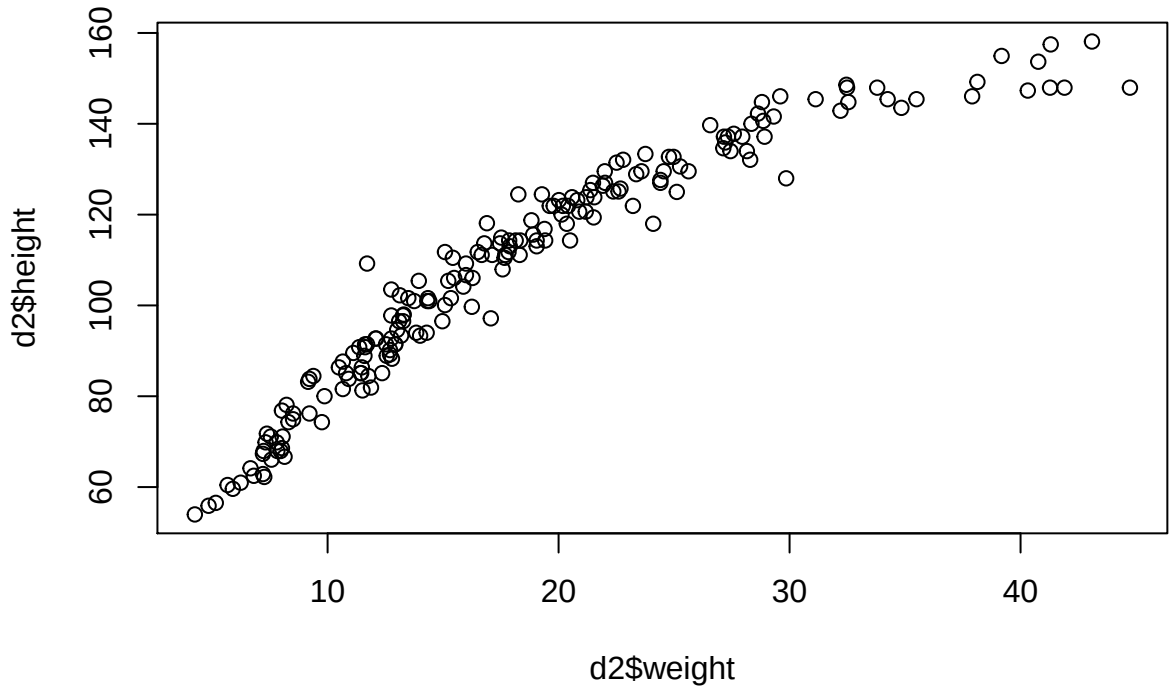
(a) Fit a linear regression to these data, using quap. Present and interpret the estimates. For every 10 units of increase in weight, how much taller does the model predict a child gets?

```
wbar = mean(d2$weight)
flist = alist(h = dnorm(mu, sigma),
              mu = a + b(w - wbar),
              alpha = 0,  # what prior do i think is reasonable for alpha?
              beta = 0,   # what prior do i think is reasonable for beta?
              sigma = 0)  # what prior do i think is reasonable for sigma?
```

```
summary(d2$weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.252  11.708  16.981  18.414  23.417  44.736
```

```
plot(d2$weight, d2$height)
```



When weight is near the mean (18 kg), then height seems to be between 90 and 120.

```
sd(d2$height)
```

```
## [1] 25.74514
```

The standard deviation of height is 25. (Plus we know it can't be negative.). We can choose a wide range here between 0 and 50 because it can't hurt... As for the prior on beta, it seems to be the case that height goes up as weight goes up, so it's reasonable to assume the beta prior should be bounded at 0.

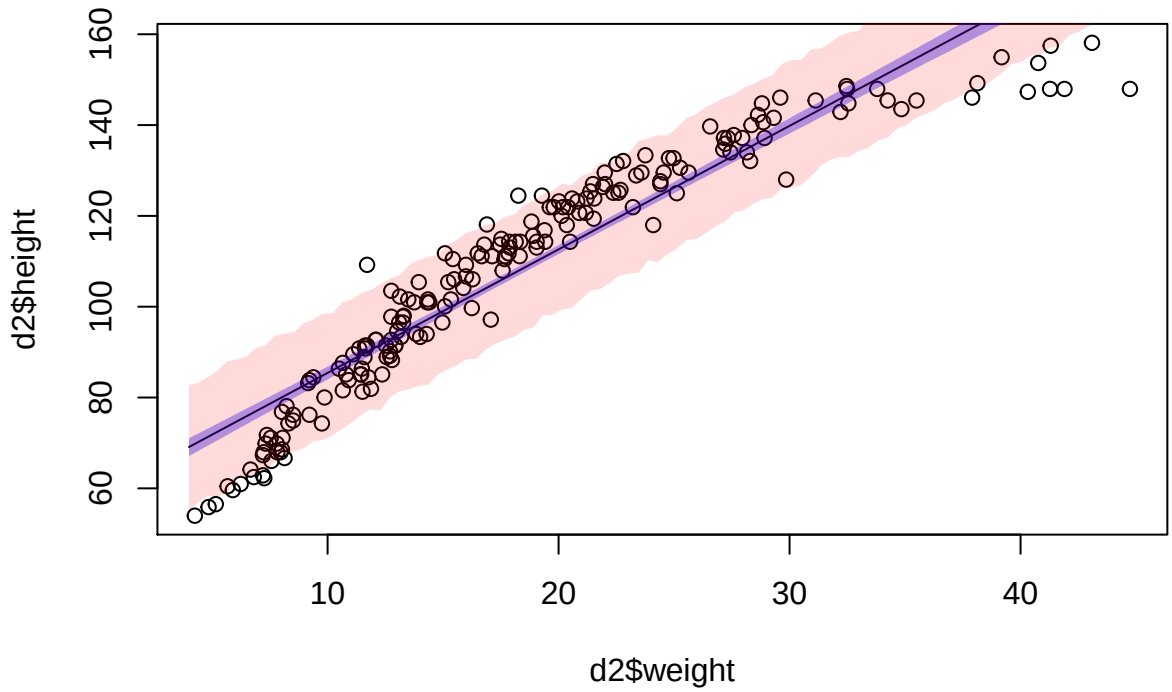
```
# i am yet unclear as to why these are bad priors...but they certainly seem to be bad
#flist = alist(height = dnorm(mu, sigma),
 #              mu = a + b*(weight - wbar),
  #             a = dunif(90, 120),  # based on plots
   #            b = dunif(0, 5),   # based on plots
    #           sigma = dunif(0, 50))  # based on direct calc from the data

flist = alist(height ~ dnorm(mu, sigma),
              mu ~ a + b*(weight - wbar),
              a ~ dnorm(105, 10),  # based on plots
              b ~ dunif(0, 5),   # based on plots
              sigma ~ dunif(0, 50))  # based on direct calc from the data

model = quap(flist, d2)
precis(model)
```

```
##             mean         sd       5.5%     94.5%
## a     108.306513 0.60777759 107.335167 109.27786
## b       2.720094 0.06829315   2.610949   2.82924
## sigma   8.437193 0.43056584   7.749066   9.12532
```

Based on these results, the model thinks that for every 10 kg increase in weight, the child would get 2.68*10 = 26.8cm cm taller. From an American standpoint, this would be like saying if a child gains 22 pounds, they will likely grow about a foot. Seems like like not *wonderful* intuition which may be a result of bad priors...

(b) Plot the raw data, with height on the vertical axis and weight on the horizontal axis. Superimpose the MAP regression line and 89% interval for the mean. Also superimpose the 89% interval for predicted heights.

```
# matrix: for each weight in the raw data, the model produces a list of heights
weight_range = seq(from = 4, to = 45, length.out = 100)

avg_mu_per_weight = link(model, data = list(weight = weight_range, n = 10000))

# compute the mean height, mu, per weight
MAP_reg_line = apply(avg_mu_per_weight, 2, mean)

# compute the 89% CI for the average height
mean_mu.PI = apply(avg_mu_per_weight, 2, PI, prob = .89)

# simulate heights for each weight
sim.height = sim(model, data = list(weight = weight_range, n = 10000))

# calculate 89% PI for the simulated heights
height.PI = apply(sim.height, 2, PI, prob = .89)


plot(d2$weight, d2$height)
lines(weight_range, MAP_reg_line) # plot the mean height per weight
shade(mean_mu.PI, weight_range, col = col.alpha('blue', .35)) # plot uncerntainty in MEAN height
shade(height.PI, weight_range, col = col.alpha('red', .15)) # plot uncertainty in HEIGHT
```



###

If you don't use a weight sequence (and pass d2$weight instead), then the shading looks ah-ful. I am not really sure why though...

(c) What aspects of the model fit concern you? Describe the kinds of assumptions you would change, if any, to improve the model. You don't have to write any new code. Just explain what the model appears to be doing a bad job of, and what you hypothesize would be a better model.

Mostly the linear fit fails to capture the behavior of this data at extremely high or low values of weight. It's not the worst linear model I've ever seen but if we need better accuracy than this, our current model misses the mark. A polynomial model or some kind of spline would be likely to improve performance.

4H3. Suppose a colleague of yours, who works on allometry, glances at the practice problems just above. Your colleague exclaims, "That's silly. Everyone knows that it's only the logarithm of body weight that scales with height!" Let's take your colleague's advice and see what happens. (a) Model the relationship between height (cm) and the natural logarithm of weight (log-kg). Use the entire Howell1 data frame, all 544 rows, adults and non-adults. Fit this model, using quadratic approximation:

```
#h_i = Normal(mu_i, sigma)
#mui = alpha + beta log(wi)
#alpha = Normal(178, 20)
#beta = Log Normal(0, 1)
#sigma  Uniform(0, 50)
```

where hi is the height of individual i and wi is the weight (in kg) of individual i. The function for computing a natural log in R is just log. Can you interpret the resulting estimates?

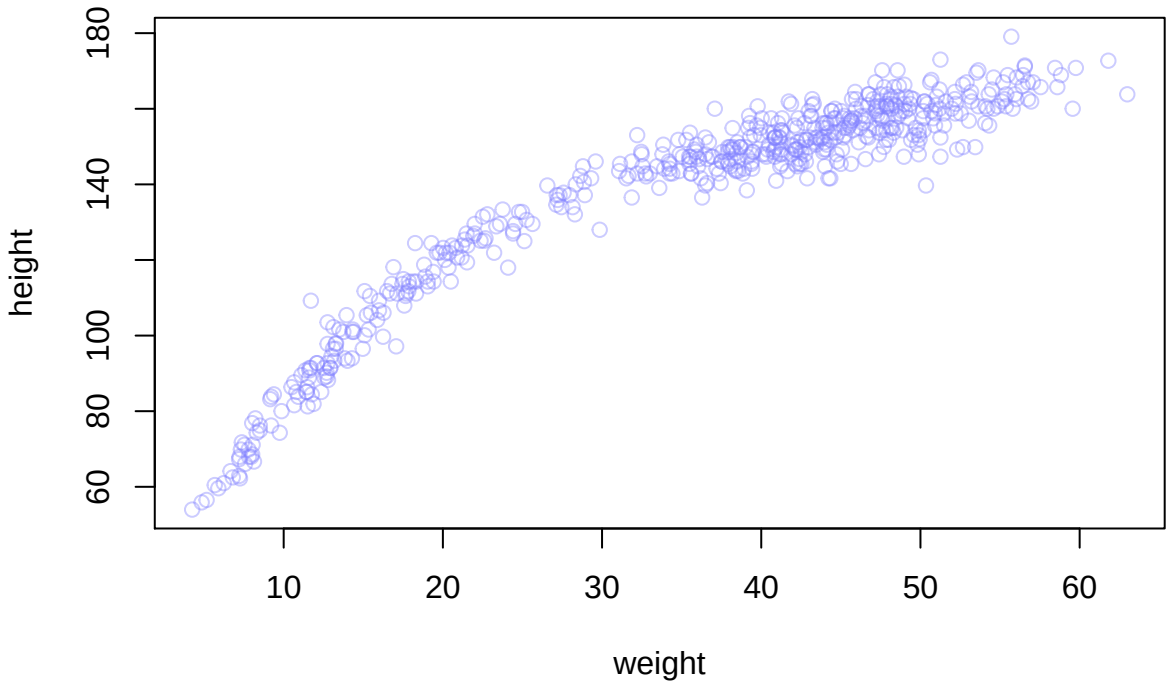
```
flist <- alist(height ~ dnorm(mu, sigma),
               mu ~ alpha + beta * log(weight),
               # could do: mu <- a + b*( log( weight ) - xbarl ), not sure this changes it much
               alpha ~ dnorm(178, 20),
               beta ~ dlnorm(0,1),
               sigma ~ dunif(0, 50)
               )

model = quap(flist, data = d)
precis(model)
```

```
##            mean        sd       5.5%      94.5%
## alpha -22.87436 1.3342915 -25.006813 -20.741901
## beta   46.81781 0.3823242  46.206778  47.428834
## sigma   5.13709 0.1558849   4.887956   5.386224
```

This is saying for every 1 unit increase in the *logged weight* of an individual, their height will increase by 46 centimeters. Transforming back to standard units for weight,

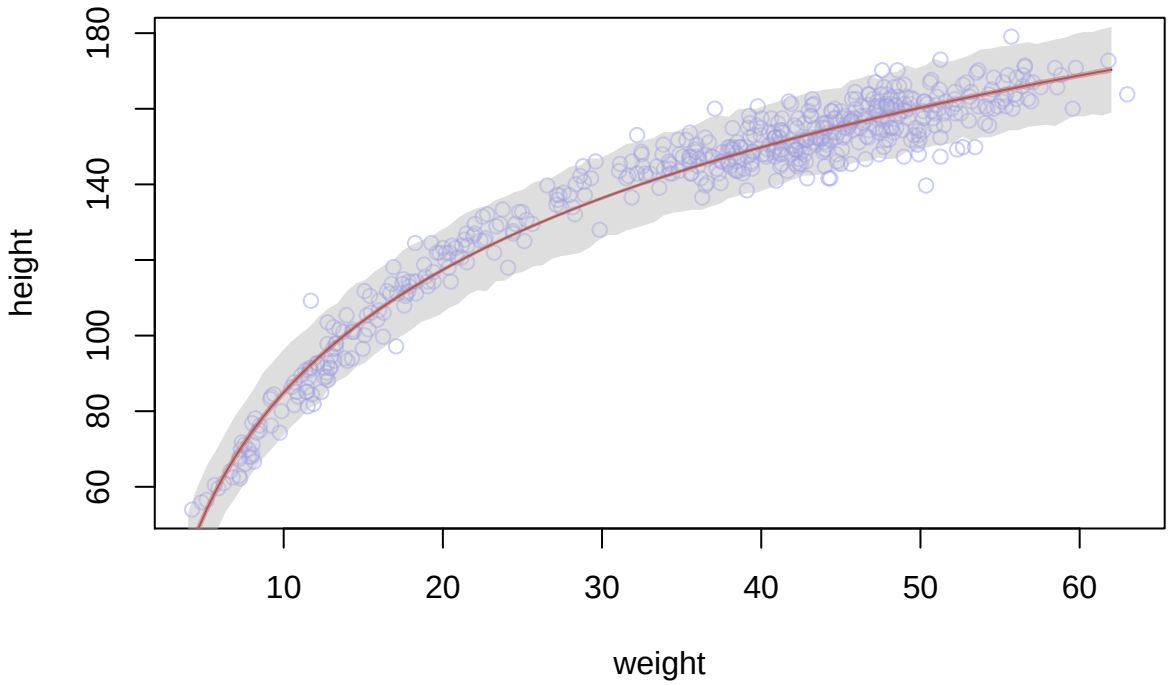
(b) Begin with this plot:

```
plot( height ~ weight , data=Howell1 ,
col=col.alpha(rangi2,0.4) )
```



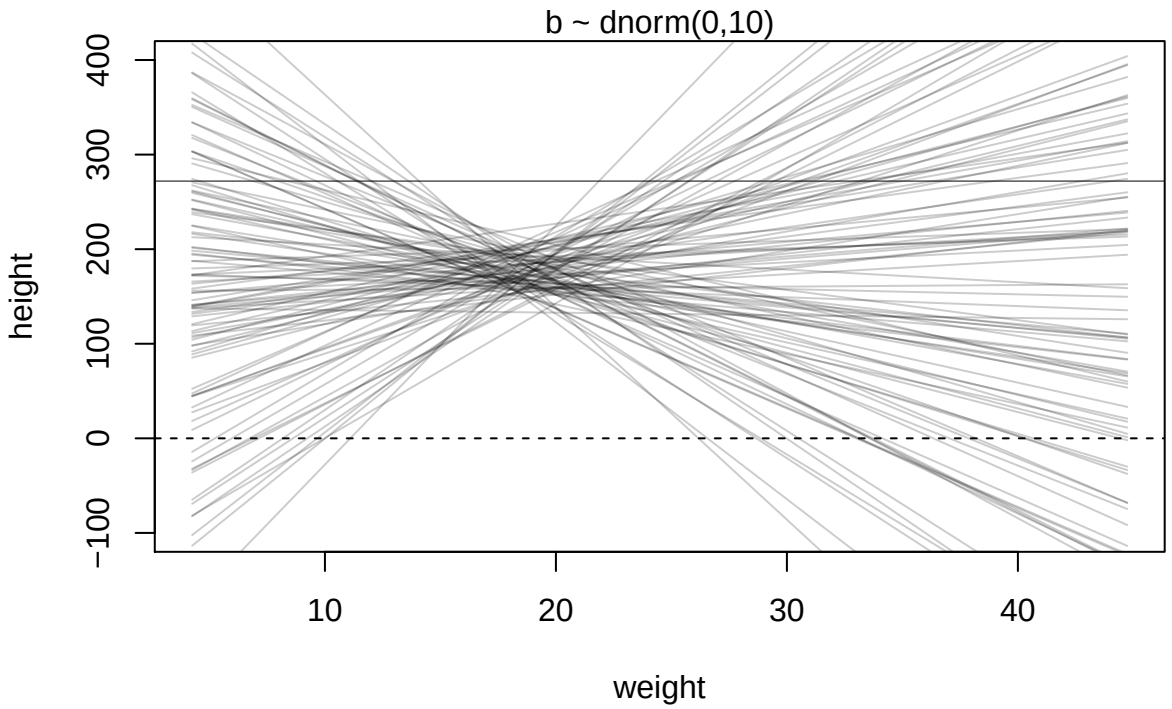
Then use samples from the quadratic approximate posterior of the model in (a) to superimpose on the plot: (1) the predicted mean height as a function of weight, (2) the 97% interval for the mean, and (3) the 97% interval for predicted heights.

```
# use link function to extract samples from the model
weight_seq = seq(from = 4, to = 62, length.out = 100)

# compute the mean height for each weight
linked_heights = link(model, data = list(weight = weight_seq))
mean_heights = apply(linked_heights, 2, mean)

mean_uncertainty = apply(linked_heights, 2, PI, prob = .97)


# simulate heights for each weight
sim_heights = sim(model, data = list(weight = weight_seq))
# compute the 89% PI for the mean
mean_heights.PI = apply(sim_heights, 2, PI, prob = .97)


plot( height ~ weight , data=d ,col=col.alpha(rangi2, 0.4) )

# predicted height as a function of weight
lines(weight_seq, mean_heights, col = 'black')

# 97% interval for the mean
shade(mean_uncertainty, weight_seq, col = col.alpha('red',  .5))

# 97% interval for predicted heights
shade(mean_heights.PI, weight_seq, col = col.alpha('grey', .5))
```



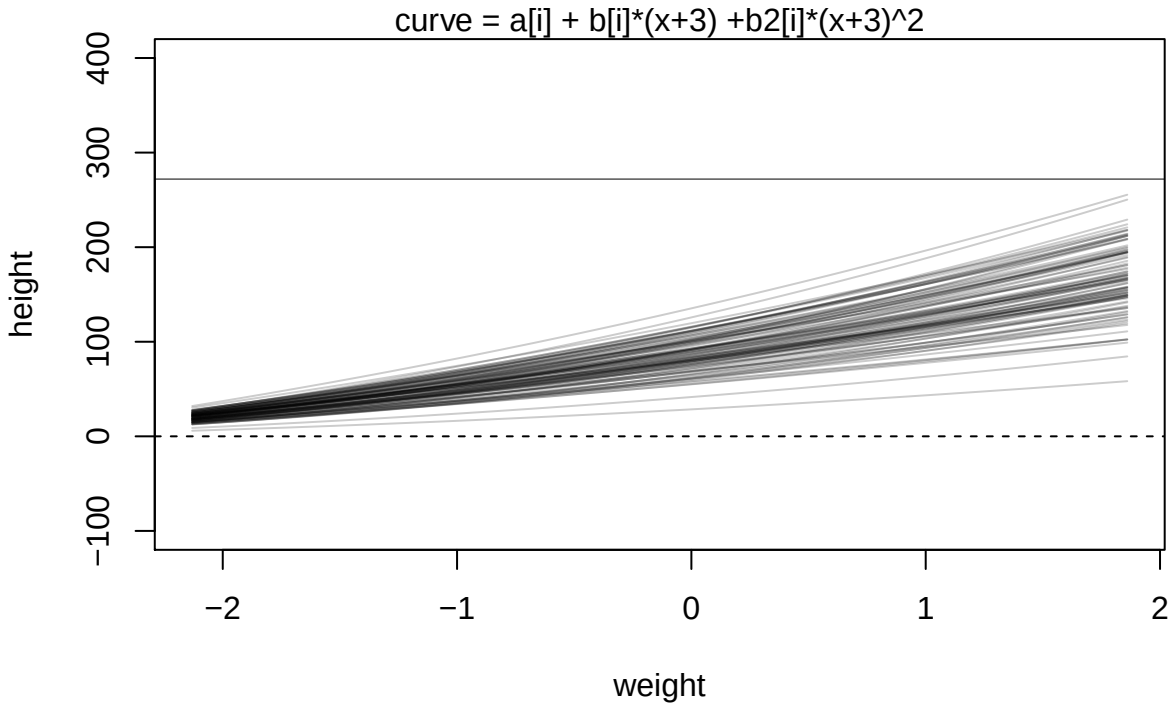
4H4. Plot the prior predictive distribution for the polynomial regression model in the chapter. You can modify the code that plots the linear regression prior predictive distribution. Can you modify the prior distributions of alpha, beta1, and beta2 so that the prior predictions stay within the biologically reasonable outcome space? That is to say: Do not try to fit the data by hand. But do try to keep the curves consistent with what you know about height and weight, before seeing these exact data.

```
set.seed(2971)
N <- 100 # 100 lines
a <- rnorm( N , 178 , 20 )
b <- rnorm( N , 0 , 10 )


plot( NULL , xlim=range(d2$weight) , ylim=c(-100,400) ,
xlab="weight" , ylab="height" )
abline( h=0 , lty=2 )
abline( h=272 , lty=1 , lwd=0.5 )
mtext( "b ~ dnorm(0,10)" )
xbar <- mean(d2$weight)
for ( i in 1:N )
  curve( a[i] + b[i]*(x - xbar) ,
         from=min(d2$weight) , to=max(d2$weight) ,
         add=TRUE , col=col.alpha("black",0.2) )
```




```
set.seed(2971)
d$w <- scale(d$weight)

N <- 100 # 100 lines
#a <- rnorm( N , 141.5 , 10 )
#b <- rnorm( N , 30.5 , 3 )
#b2 <- rnorm(N, -7.625, 2) # height grows "upside down parabola-like as weight increases"

a <- rexp( N , 1) # when weight is 0, height is 20 cm
b <- rnorm( N , 20 , 5 )
b2 <- rnorm(N, 3, 1)
#b3 <- rnorm(N, 0, .1) # why are we getting rid of b3, because it makes it that much harder right?
```

```
plot( NULL , xlim=range(d$w) , ylim=c(-100,400) ,
xlab="weight" , ylab="height" )
abline( h=0 , lty=2 )
abline( h=272 , lty=1 , lwd=0.5 )
mtext( "curve = a[i] + b[i]*(x+3) +b2[i]*(x+3)^2" )


for ( i in 1:N )
  curve( a[i] + b[i]*(x+3) +b2[i]*(x+3)^2,
         from=min(d$w) , to=max(d$w) ,
         add=TRUE , col=col.alpha("black",0.2) )
```


curve = a[i] + b[i]*(x+3) +b2[i]*(x+3)^2
height
400
300
200
100
0
−100
−2
−1
0
1
2
weight