hw2

GISH

2019撟3<9c><88>15

## Question 1

By using the Howell1 data mentioned in the class, the weights listed below were recorded in the !Kung census, but heights were not recorded for these individuals. Provide predicted heights and 89% intervals (either HPDI or PI) for each of these individuals. That is, fill in the table below, using model-based predictions.

library(rethinking)

## Loading required package: rstan

## Loading required package: ggplot2

## Loading required package: StanHeaders

## rstan (Version 2.18.2, 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)

## For improved execution time, we recommend calling  
## Sys.setenv(LOCAL\_CPPFLAGS = '-march=native')  
## although this causes Stan to throw an error on a few processors.

## Loading required package: parallel

## rethinking (Version 1.59)

library(tidybayes)

## NOTE: As of tidybayes version 1.0, several functions, arguments, and output column names  
## have undergone significant name changes in order to adopt a unified naming scheme.  
## See help('tidybayes-deprecated') for more information.

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

data(Howell1)  
  
d <- Howell1  
d$weight\_st <- (d$weight - mean(d$weight)) / sd(d$weight)  
model <- map(  
 alist(  
 height ~ dnorm(mu, sigma),  
 mu <- alpha + beta\*weight\_st,  
 alpha ~ dnorm(178,20),  
 beta ~ dlnorm(0,1),  
 sigma ~ dunif(0,50)  
 ),  
 data <- d  
)  
  
  
some\_weights <- c(46.95, 43.72, 64.78, 32.59, 54.63)  
some\_weights\_standard <- (some\_weights - mean(d$weight)) / sd(d$weight)  
simulated\_heights <- sim(model, data = list(weight\_st = some\_weights\_standard))

## [ 100 / 1000 ]  
[ 200 / 1000 ]  
[ 300 / 1000 ]  
[ 400 / 1000 ]  
[ 500 / 1000 ]  
[ 600 / 1000 ]  
[ 700 / 1000 ]  
[ 800 / 1000 ]  
[ 900 / 1000 ]  
[ 1000 / 1000 ]

simulated\_heights\_mean <- apply(X = simulated\_heights, MARGIN = 2, FUN = mean)  
  
samples <- extract.samples(model)  
  
  
individualone <- rnorm(n = 10000, mean = samples$alpha + samples$beta\*some\_weights\_standard[1], sd = samples$sigma)  
individualone\_mean <- mean(individualone)  
individualone\_hdi <- mode\_hdi(individualone, .width=0.89)  
  
individualtwo <- rnorm(n = 10000, mean = samples$alpha + samples$beta\*some\_weights\_standard[2], sd = samples$sigma)  
individualtwo\_mean <- mean(individualtwo)  
individualtwo\_hdi <- mode\_hdi(individualtwo, .width=0.89)  
  
individualthree <- rnorm(n = 10000, mean = samples$alpha + samples$beta\*some\_weights\_standard[3], sd = samples$sigma)  
individualthree\_mean <- mean(individualthree)  
individualthree\_hdi <- mode\_hdi(individualthree, .width=0.89)  
  
individualfour <- rnorm(n = 10000, mean = samples$alpha + samples$beta\*some\_weights\_standard[4], sd = samples$sigma)  
individualfour\_mean <- mean(individualfour)  
individualfour\_hdi <- mode\_hdi(individualfour, .width=0.89)  
  
individualfive <- rnorm(n = 10000, mean = samples$alpha + samples$beta\*some\_weights\_standard[5], sd = samples$sigma)  
individualfive\_mean <- mean(individualfive)  
individualfive\_hdi <- mode\_hdi(individualfive, .width=0.89)  
  
listA = c(individualone\_mean,individualtwo\_mean ,individualthree\_mean ,individualfour\_mean ,individualfive\_mean)  
listB = c(individualone\_hdi$ymin,individualtwo\_hdi$ymin,individualthree\_hdi$ymin,individualfour\_hdi$ymin,individualfive\_hdi$ymin)  
listC = c(individualone\_hdi$ymax,individualtwo\_hdi$ymax,individualthree\_hdi$ymax,individualfour\_hdi$ymax,individualfive\_hdi$ymax)  
  
answer.sheet = tibble(Individual = 1:5, Weight = some\_weights, Expected\_height = listA, lowerbound89hdpi = listB, upperbound89hdpi = listC)  
  
answer.sheet

## # A tibble: 5 x 5  
## Individual Weight Expected\_height lowerbound89hdpi upperbound89hdpi  
## <int> <dbl> <dbl> <dbl> <dbl>  
## 1 1 47.0 158. 144. 174.  
## 2 2 43.7 152. 137. 167.  
## 3 3 64.8 190. 174. 204.  
## 4 4 32.6 133. 118. 148.  
## 5 5 54.6 172. 157. 187.

## Question 2

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

d2 = d %>% filter(age < 18)  
  
modelB <- map(  
 alist(  
 height ~ dnorm(mu, sigma),  
 mu <- alpha + beta\*d2$weight,  
 alpha ~ dnorm(100,50),  
 beta ~ dlnorm(0,1),  
 sigma ~ dunif(0,50)  
 ),  
 data <- d2  
)  
#For every 10 units of increase in weight, how much taller does the model predict a child gets?  
precis(modelB)

## Mean StdDev 5.5% 94.5%  
## alpha 58.33 1.40 56.09 60.56  
## beta 2.72 0.07 2.61 2.82  
## sigma 8.44 0.43 7.75 9.13

We can see that the mean of beta is 2.72,that means on average, if there’s 10 units of increase in weight a child gets 2.72\* ln(10) ~= 6.263 taller

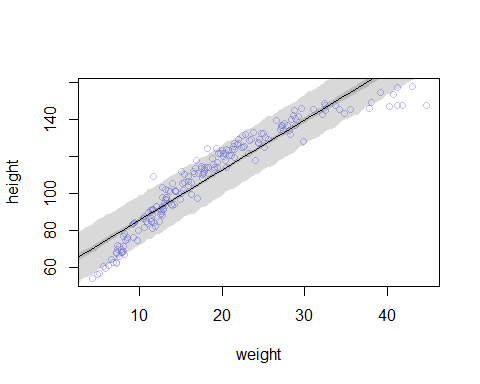
1. Plot the raw data, with height on the vertical axis and weight on the horizontal axis. Superimpose the stan predicted regression line and 89% HPDI for the mean. Also superimpose the 89% HPDI for predicted heights.

library(MASS)

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

weight\_seq <- seq(from = 0 , to = 100, by = 0.5)  
  
#makes mean  
twoBsamples <- data.frame( mvrnorm(n = 10000, coef(modelB), vcov(modelB)) )  
simMu\_func <- function(w){  
 twoBsamples$alpha + twoBsamples$beta \* w  
}  
simMu <- sapply(X = weight\_seq, FUN = simMu\_func)  
simMu\_mean <- apply(X = simMu, MARGIN = 2, FUN = mean)  
simMu\_hpdi <- apply(X = simMu, MARGIN = 2, FUN = HPDI, prob = 0.89)  
  
#makes heights  
height\_func <- function(w){  
 rnorm(n = nrow(twoBsamples), mean = simMu\_func(w), sd = twoBsamples$sigma)  
}  
height\_samples <- sapply(X = weight\_seq, FUN = height\_func)  
height\_hpdi <- apply(X = height\_samples, MARGIN = 2, FUN = HPDI, prob = .89)  
  
#plot  
plot(height ~ weight, data = d2, col = col.alpha(rangi2, .5))  
lines(x = weight\_seq, y = simMu\_mean)  
shade(object = simMu\_hpdi, lim = weight\_seq)  
shade(object = height\_hpdi, lim = weight\_seq)

 (c) What aspects of the model fit concern you? Describe the kinds of assumption you would change, if any, to improve your 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

A: Because the data in the plot doesn’t seem to follow a straight line. First, weights seem to increase height faster, and at around 40 weight, the height gets more fixed. And we can see that as weight grows, the height’s increase are slowing down, that’s diminishing marginal increase. So we know the slope of the regression should not be a constant, and by diminishing increase, we hypothesize it to be a height = ln(weight) model.

Question 3. Suppose a colleague of yours, who works on allometry, glances at the practice problems just above. (In Question 2). You 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.

1. 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 MCMC (stan)

d3 <- Howell1  
modelC <- map(  
 alist(  
 height ~ dnorm(mean = mu, sd = sigma),  
 mu <- alpha + beta\*log(weight),  
 alpha ~ dnorm(mean = 178, sd = 100),  
 beta ~ dnorm(mean = 0, sd = 100),  
 sigma ~ dunif(min = 0, max = 50)  
 ),  
 data = d3  
)  
  
weight\_seq <- seq(from = 5 , to = 100, by = 0.5)  
  
# makes mean  
twoCsamples <- data.frame( mvrnorm(n = 10000, coef(modelC), vcov(modelC)) )  
simMu\_func <- function(w){  
 twoCsamples$alpha + twoCsamples$beta \* log(w)  
}  
simMu <- sapply(X = weight\_seq, FUN = simMu\_func)  
simMu\_mean <- apply(X = simMu, MARGIN = 2, FUN = mean)  
simMu\_hpdi <- apply(X = simMu, MARGIN = 2, FUN = HPDI, prob = 0.97)  
  
# makes heights  
height\_func <- function(w){  
 rnorm(n = nrow(twoCsamples), mean = simMu\_func(w), sd = twoCsamples$sigma)  
}  
height\_samples <- sapply(X = weight\_seq, FUN = height\_func)  
height\_hpdi <- apply(X = height\_samples, MARGIN = 2, FUN = HPDI, prob = 0.97)  
  
summary(simMu\_mean)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 51.98 134.32 162.67 153.19 180.24 193.00

Can you interpret the resulting estimates?

First, the mean of alpha is -23.78,that means on average, if a person is 1 weight(which is impossible), the person would have -23.78 height on average. And beta on average is 47.08, that means on average if a person have 10 weight, the height should be -23.78+47.08*ln(10) ~= 84.626. Seems very logical based on fact. As the weight heavily increases, for example, 100 weight, we have average height = -23.78+47.08*ln(100) ~= 193.031. We can see the increase from weight 1 to 10 (9 increase) is 84.626 – (-23.78) = 108.406. But when weight increase from 10 to 100(90 increase), the increase in height is 193.031 -84.626 = 108.405, the same. Finally, there’s sigma, the deviation from average mean of height, is on average 5.13 , and it is based on uniform distribution , so

1. Begin with this plot: plot(height~weight, data=Howell1, col=col.alpha(rangi2, 0.4)) Then use samples from the 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% HPDI for the mean, and (3) the 97% HPDI for predicted heights.

#plot  
plot(height ~ weight, data = d3, col = col.alpha(rangi2, 0.4))  
lines(x = weight\_seq, y = simMu\_mean)  
shade(object = simMu\_hpdi, lim = weight\_seq)  
shade(object = height\_hpdi, lim = weight\_seq)

