R Notebook

Theory

Part 1

Import and inspect the data

```
library(rethinking)
```

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

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

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

```
## Warning: package 'StanHeaders' was built under R version 3.5.2
```

```
## 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)
```

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

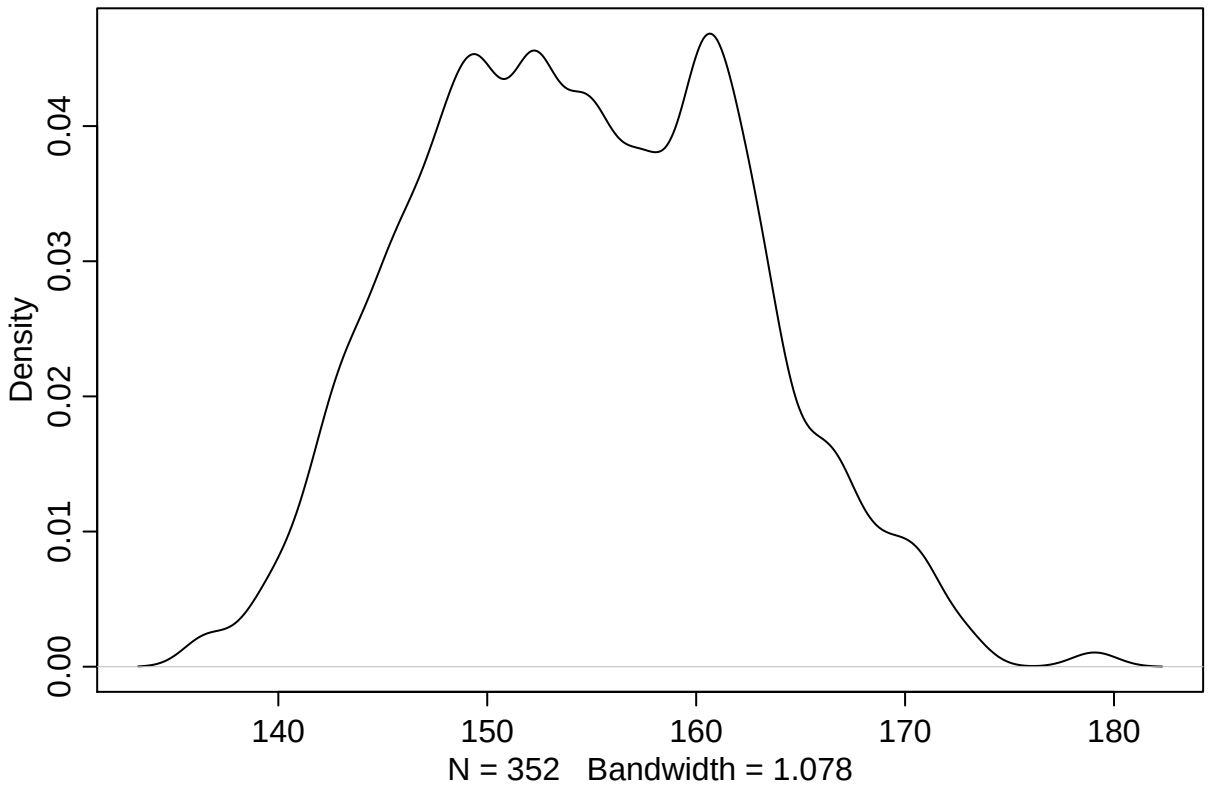
```
## rethinking (Version 1.59)
```

```
data(Howell1)
df <-  Howell1
head(df)
```

```
##    height   weight age male
## 1 151.765 47.82561  63    1
## 2 139.700 36.48581  63    0
## 3 136.525 31.86484  65    0
## 4 156.845 53.04191  41    1
## 5 145.415 41.27687  51    0
## 6 163.830 62.99259  35    1
```

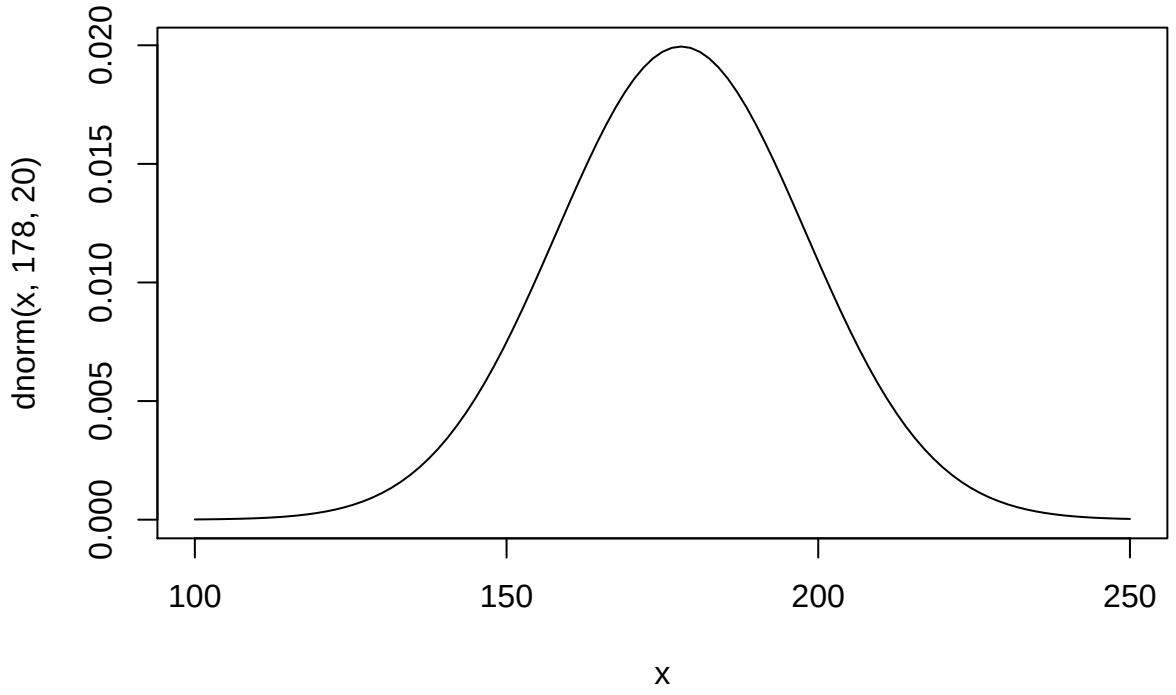
Select only adults and plot

```
df2 <- df[df$age >= 18,]
dens(df2$height)
```

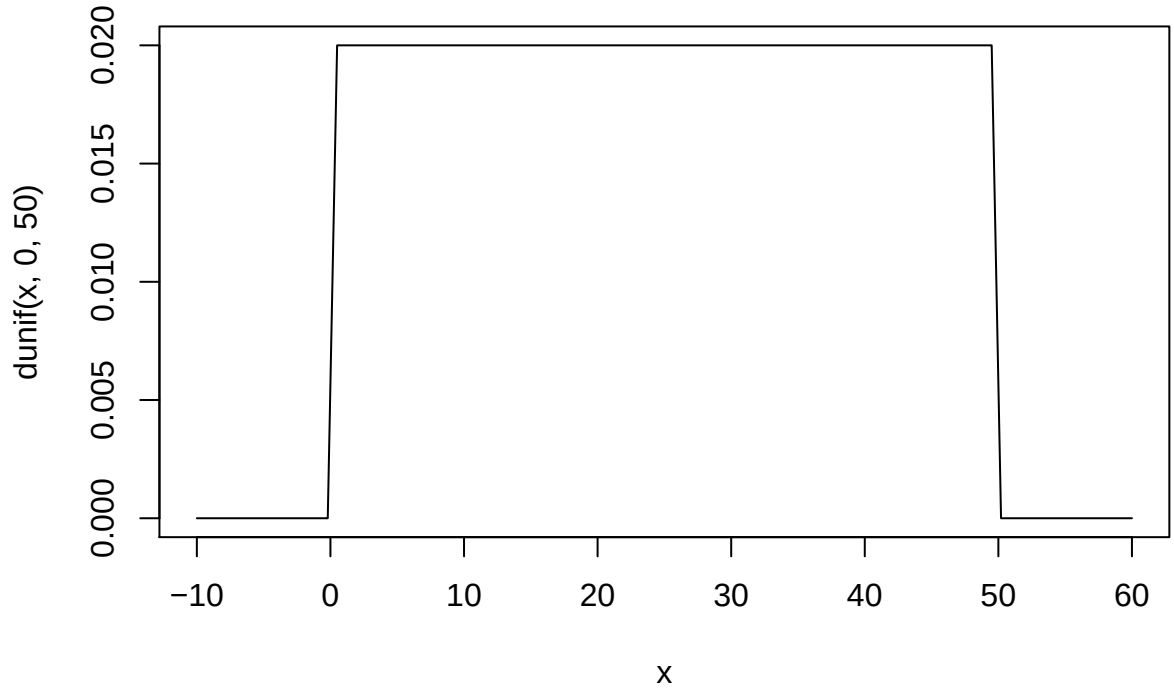


Have a look at the priors for the height and variance

```
curve( dnorm( x, 178, 20), from=100, to=250)
```

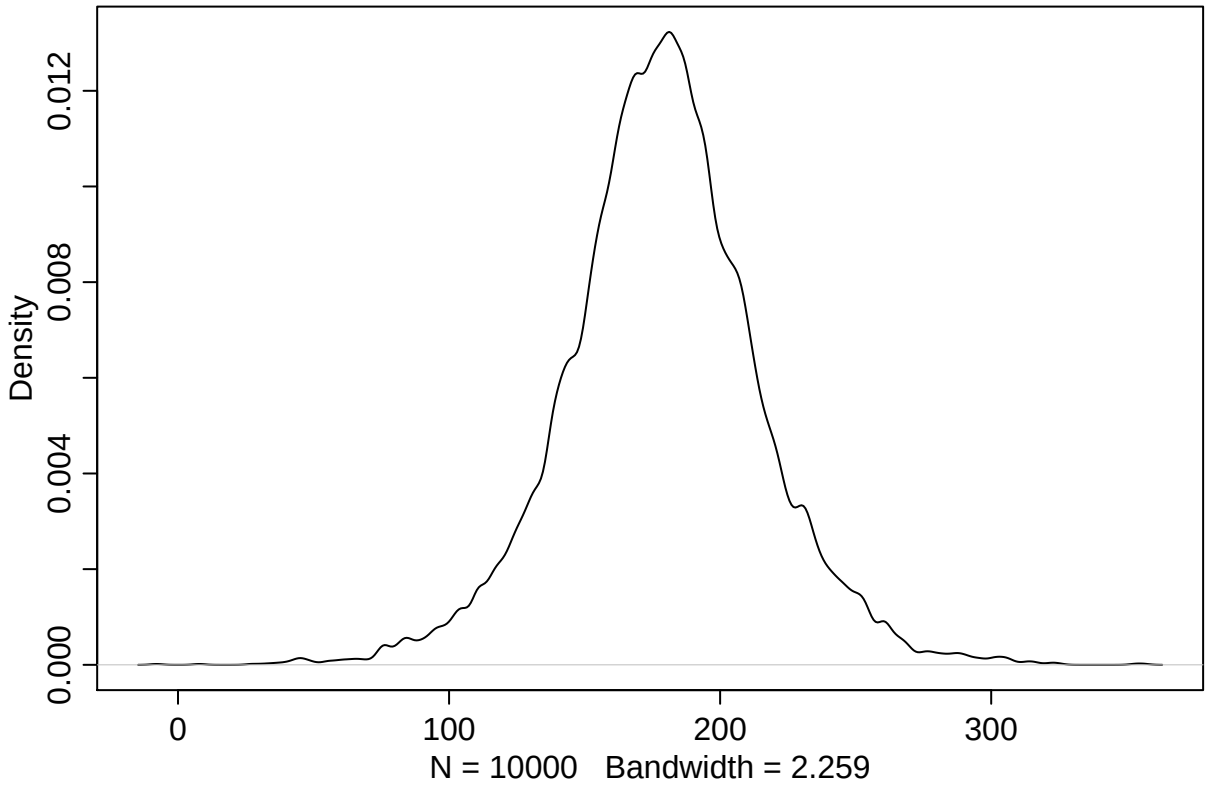



```
curve( dunif( x, 0, 50), from=-10, to=60)
```



Sample the priors for the height

```
sample_mu <- rnorm( 1e4, 178, 20)
sample_sigma <- runif( 1e4, 0, 50)
prior_h <- rnorm( 1e4, sample_mu, sample_sigma)
dens( prior_h)
```

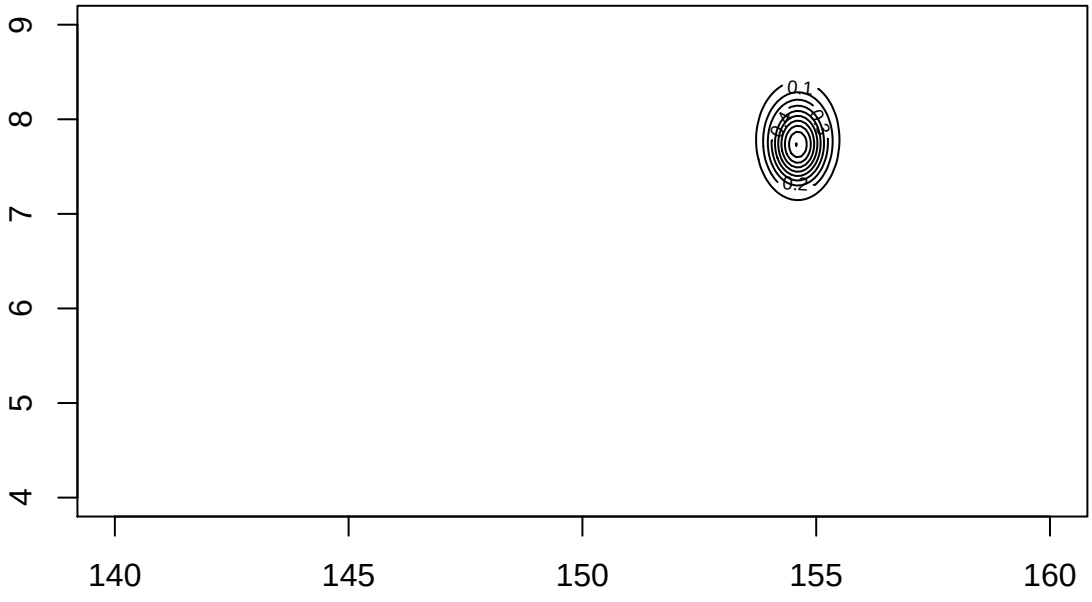


Estimate the posterior. Note that the range of mu.list and sigma.list is based on knowing the outcome from the posterior. Then the posterior probability for each point in the grid is caluclated.

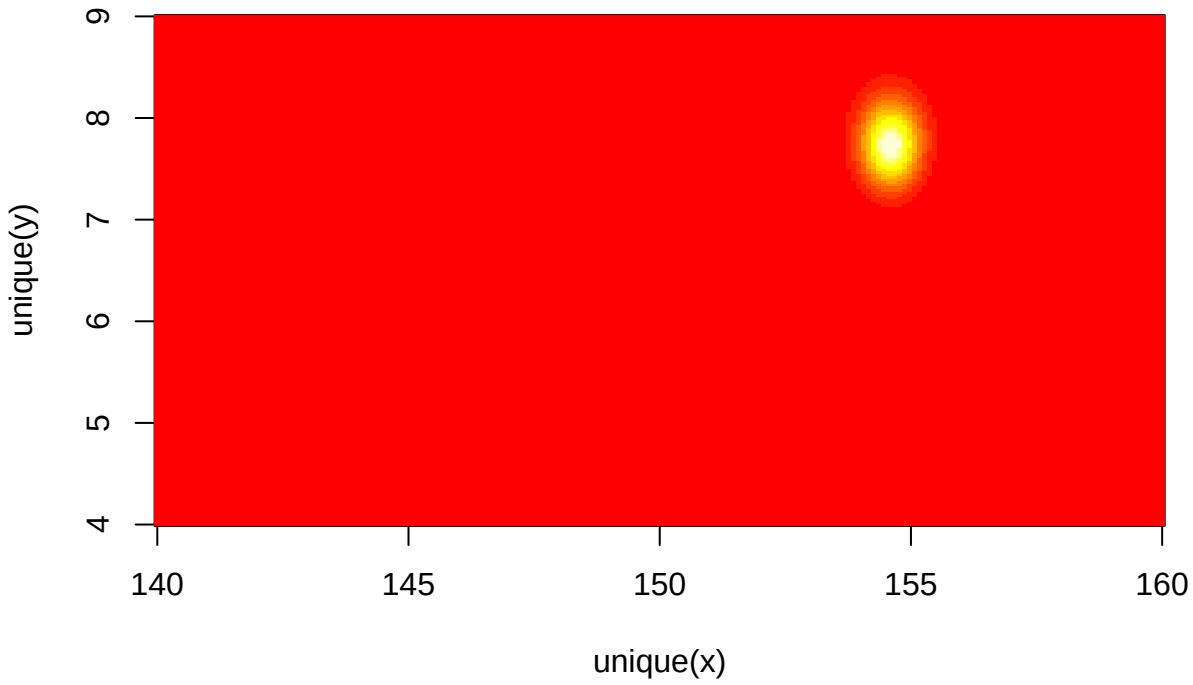
```
mu.list <- seq( from=140, to=160, length.out=200)
sigma.list <- seq( from=4, to=9, length.out=200)
post <- expand.grid( mu=mu.list, sigma=sigma.list)

post$LL <- sapply( 1:nrow(post), function(i) sum( dnorm(
  df2$height,
  mean=post$mu[i],
  sd=post$sigma[i], log=TRUE)))
post$prod <- post$LL +
  dnorm( post$mu, 178, 20, TRUE) +
  dunif( post$sigma, 0, 50, TRUE)
post$prob <- exp( post$prod - max(post$prod))
```

A quick plot shows the probability of different heights and variances.

```
contour_xyz( post$mu, post$sigma, post$prob)
```




```
image_xyz( post$mu, post$sigma, post$prob)
```



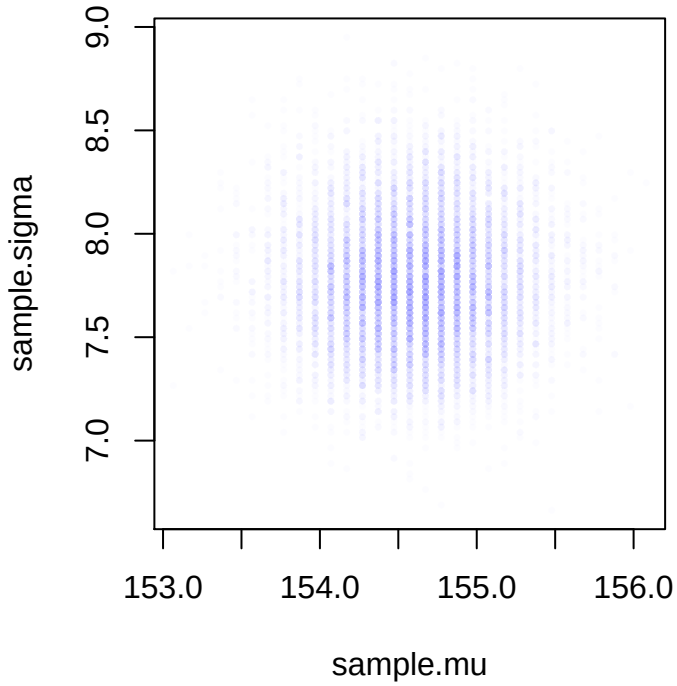
Adults are expected to be around 155 cm tall.

Sampeling from the posterior is an alternative way of identifying the same pattern. This is the more flexible approach.

```
sample.rows <- sample( 1:nrow(post), size=1e4, replace=TRUE, prob=post$prob)
sample.mu <- post$mu[ sample.rows]
sample.sigma <- post$sigma[ sample.rows]
```

Plotting the samples with alpha less than 1.

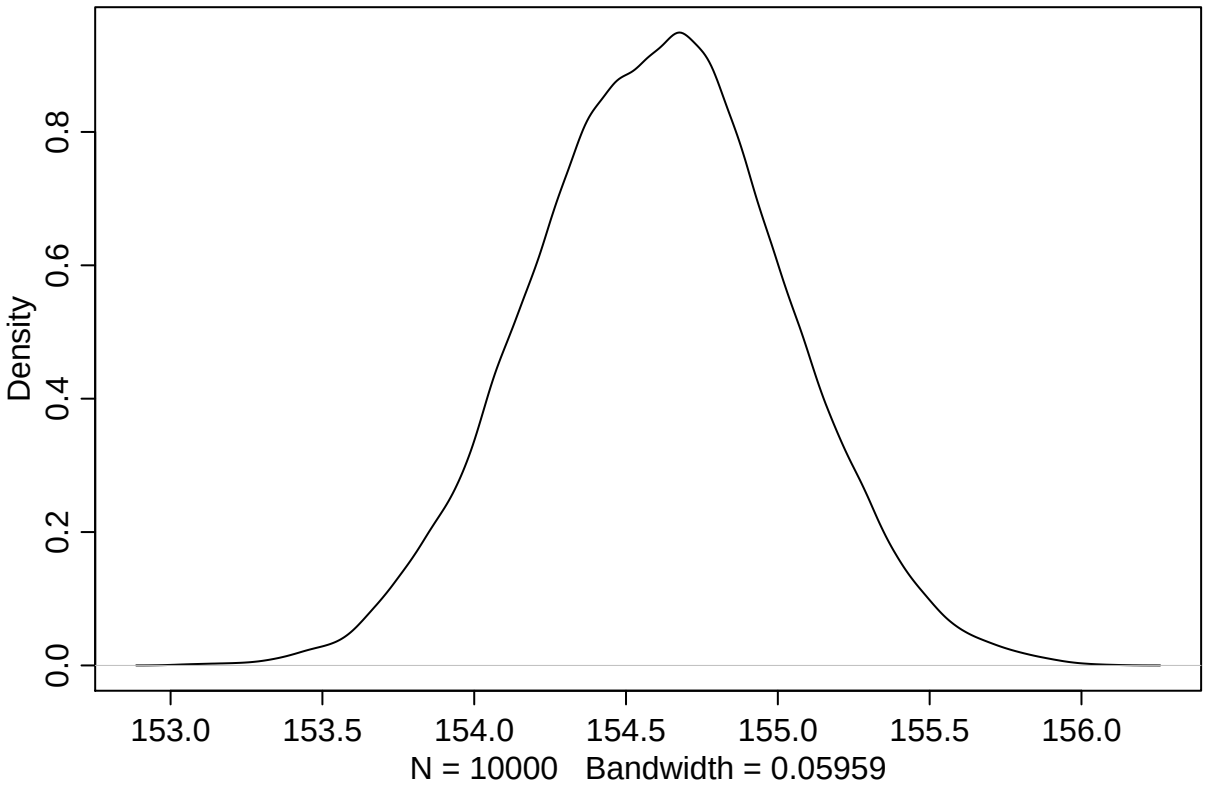
```
par(pty="s")
plot( sample.mu, sample.sigma, cex=0.5, pch=16, col=col.alpha(rangi2,0.02))
```



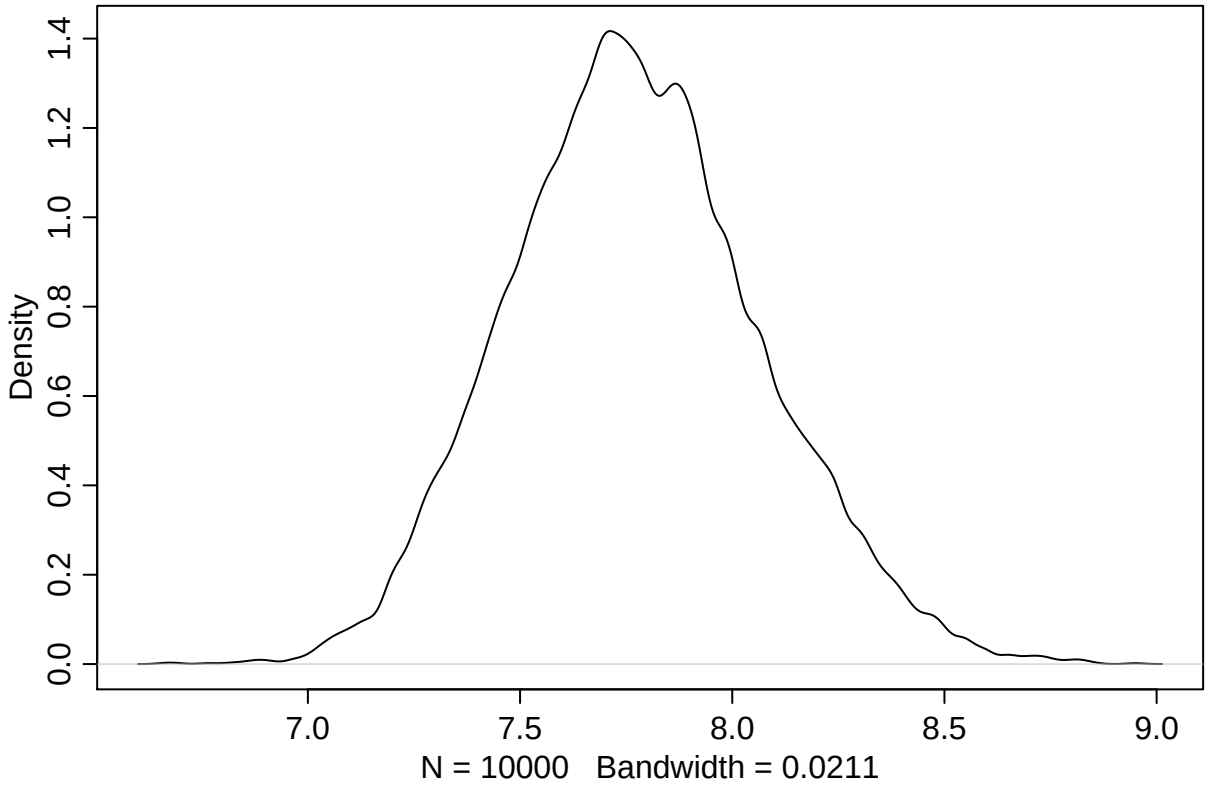
This shows that simulations from the posterior gives the same visual impression as the calulated posterior.

Some other ways the posterior samples can be described:

```
dens( sample.mu, adj=1)
```



```
dens( sample.sigma)
```




```
HPDI( sample.mu)
```

```
##    |0.89    0.89|
## 153.8693 155.1759
```

```
HPDI( sample.sigma)
```

```
##    |0.89    0.89|
## 7.291457 8.221106
```

Part 2

```
rm(list=ls())
```

Loading the data and selecting only adults.

```
library(rethinking)
data(Howell1)
df <- Howell1
df2 <- df[ df$age >= 18,]
```

Define the model and find the maximum a posteriori estimates

```
flist <- alist(
  height ~ dnorm( mu, sigma),
  mu ~ dnorm( 178, 20),
  sigma ~ dunif( 0, 50)
```

```
  )
m4.1 <- map( flist, data=df2)
```

Summ ary of the fitted moddel

```
precis(m4.1)
```

```
##         Mean StdDev   5.5%  94.5%
## mu    154.61   0.41 153.95 155.27
## sigma   7.73   0.29   7.27   8.20
```

```
vcov(m4.1)
```

```
##                 mu        sigma
## mu    0.1697742767 0.0002198558
## sigma 0.0002198558 0.0849491921
```

```
cov2cor(vcov(m4.1))
```

```
##                mu       sigma
## mu    1.000000000 0.001830722
## sigma 0.001830722 1.000000000
```

Part 3, adding a predictor

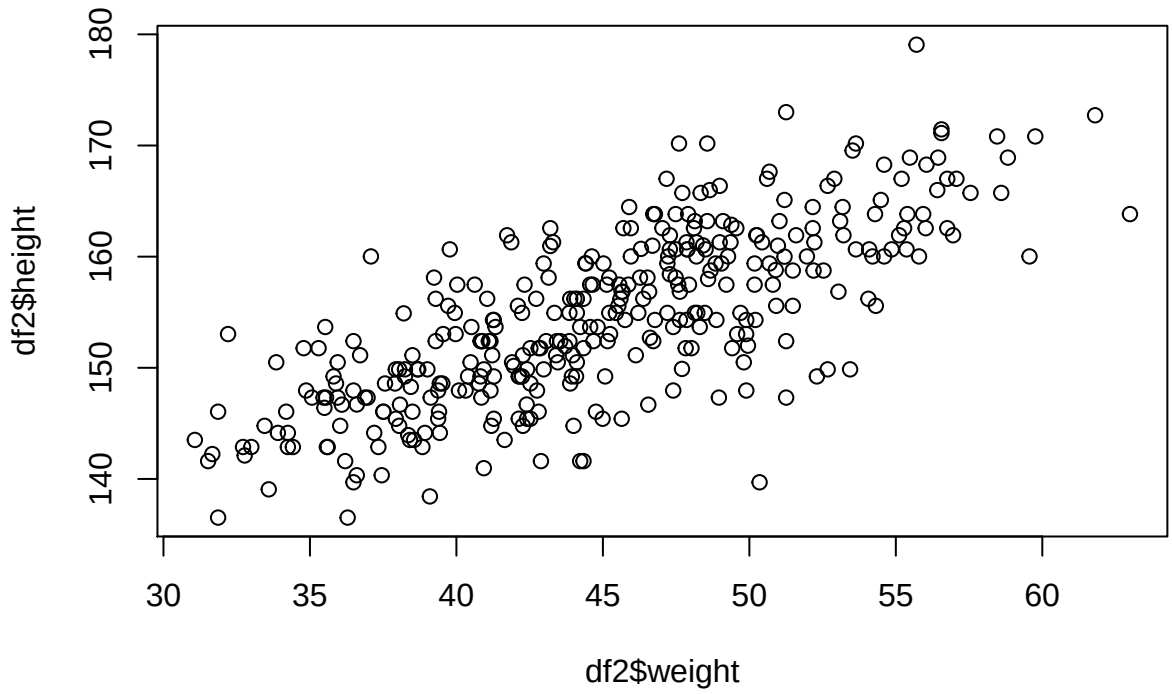
```
rm(list=ls())
```

Load and select the data

```
library(rethinking)
data(Howell1)
df <- Howell1
df2 <- df[ df$age >= 18,]
```

A quick visual inspection

```
plot(df2$height ~ df2$weight)
```



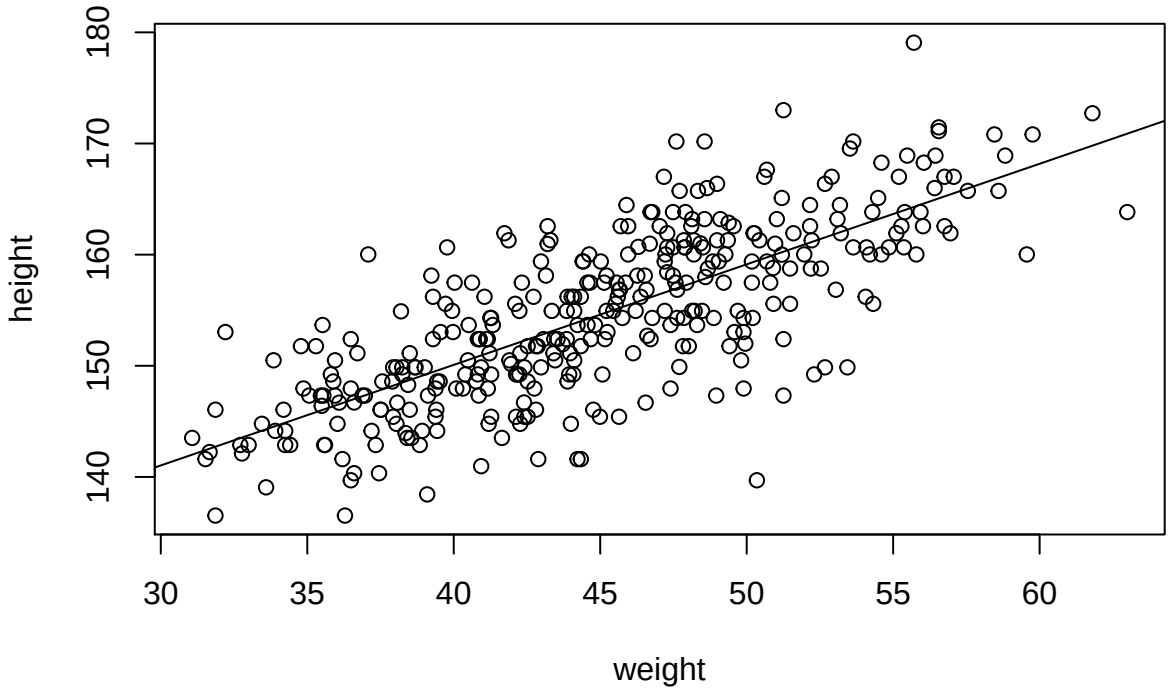
Define and fit the model. Then inspect it

```
m4.3 <- map( alist( height ~ dnorm( mu, sigma),
                    mu <- a + b*weight,
                    a ~ dnorm( 156, 100),
                    b ~ dnorm( 0, 10),
                    sigma ~ dunif( 0, 50) ) ,
             data=df2)
m4.3
```

```
##
## Maximum a posteriori (MAP) model fit
##
## Formula:
## height ~ dnorm(mu, sigma)
## mu <- a + b * weight
## a ~ dnorm(156, 100)
## b ~ dnorm(0, 10)
## sigma ~ dunif(0, 50)
##
## MAP values:
##           a           b       sigma
## 113.8954178   0.9046799   5.0718554
##
## Log-likelihood: -1071.01
```

A plot with the line from the MAP estimated parameters s

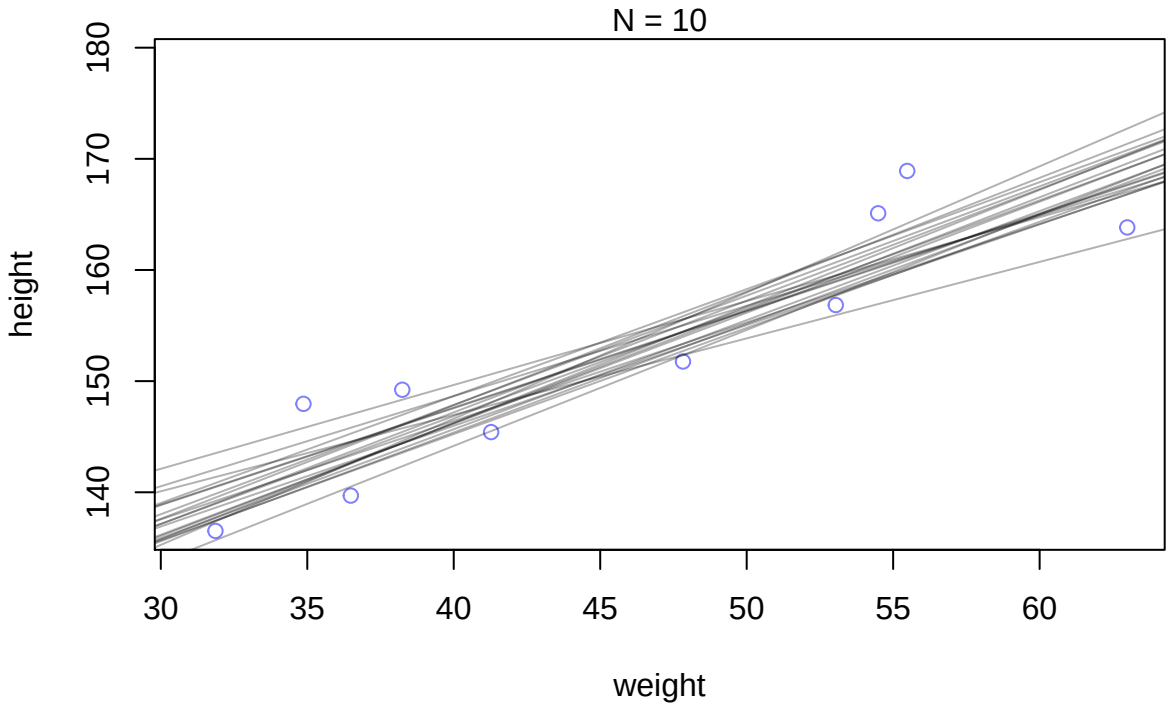
```
plot( height ~ weight, data=df2)
abline( a=coef(m4.3)["a"], b=coef(m4.3)["b"])
```



The estimated paramters have a distribution (joint probability distribution). Selecting 10 samples and estimating the parameters to show the inner workings of the model.

```
N <- 10
dfN <- df2[ 1:N,]
mN <- map( alist( height ~ dnorm( mu, sigma),
                  mu <- a + b*weight,
                  a ~ dnorm( 178, 100),
                  b ~ dnorm( 0, 10),
                  sigma ~ dunif( 0, 50) ), data=dfN)
```

```
#extract 20 samples from the posterior
post <- extract.samples( mN, n=20)
#display raw data and sample size
plot( dfN$weight,
      dfN$height,
      xlim=range(df2$weight),
      ylim=range(df2$height),
      col=rangi2,
      xlab="weight",
      ylab="height")
mtext(concat("N = ",N))
#plot the lines, with transparency
for (i in 1:20)
  abline( a=post$a[i], b=post$b[i], col=col.alpha("black",0.3))
```



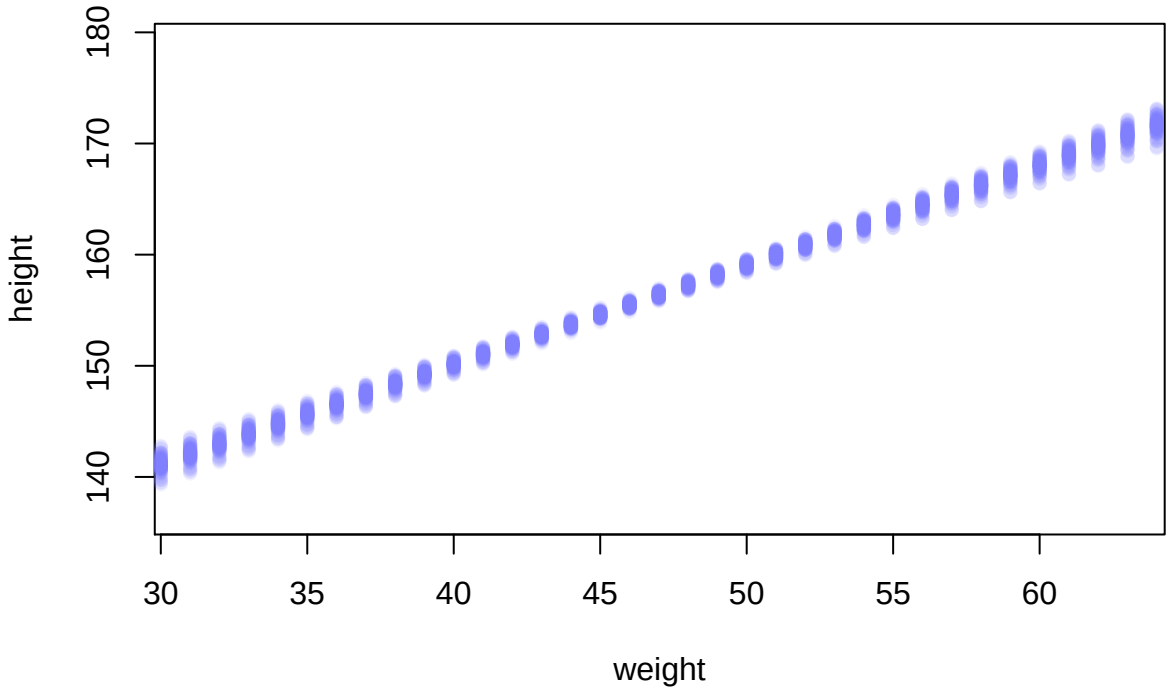
Link is a function that can be used to get the simulated expected model value conditional on the weight. This can be used to make a similar visualization to the one using 10 data points and 20 simulations but with all the data.

```
weight.seq=seq(25,70, by = 1)
mu <- link( m4.3, data=data.frame(weight=weight.seq))
```

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

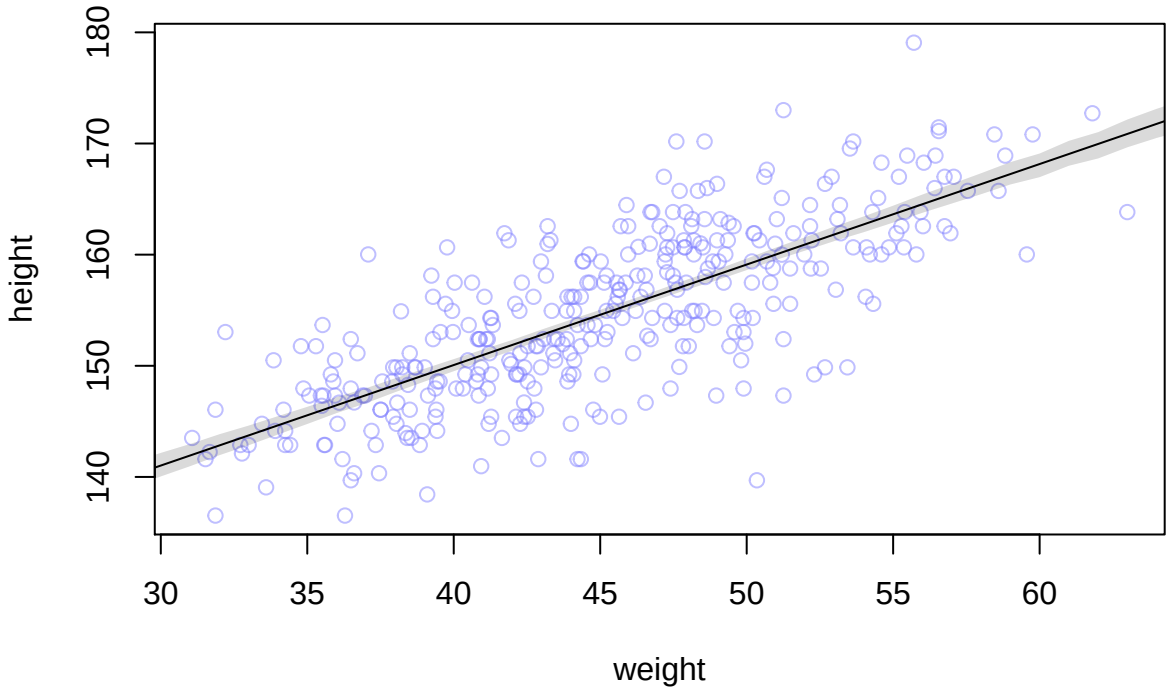
Plotting the different mu values

```
#use type="n" to hide raw data, blank plot with axis
plot( height ~ weight, df2, type="n")

#loop over the first 100 sample rows and plot the mu value coresponding to each weight
for (i in 1:100)
  points( weight.seq, mu[i,], pch=16, col=col.alpha(rangi2,0.1))
```



The simulateded mu values can be summarised and then plotted

```
#summarize the distribution of mu
mu.mean <- apply( mu, 2, mean)
mu.HPDI <- apply( mu, 2, HPDI, prob=0.89)
#plot raw data
#fading out points to make line and interval more visible
plot( height ~ weight, data=df2, col=col.alpha(rangi2,0.5))
#plot the MAP line, aka the mean mu for each weight
lines( weight.seq, mu.mean)
#plot a shaded region for 89% HPDI
shade( mu.HPDI, weight.seq)
```



Easy

```
rm(list=ls())
```

Question 1

The first line

Question 2

Two parameters

Question 3

$$\mathrm{P}(\mu, \sigma \mid y) = \frac{\mathrm{P}(y \mid \mu, \sigma) \times \mathrm{P}(\mu, \sigma)}{\mathrm{P}(y)}$$

The priors of μ and σ are independent. The joint prior is a product of their individual priors. Let f be the normal distribution.

$$\mathrm{P}(\mu, \sigma \mid y) = \frac{\prod f(y \mid \mu, \sigma) \times f(\mu \mid 0, 10) \times 1/10}{\int_\mu \int_\sigma \prod f(y \mid \mu, \sigma)}$$

Question 4

line 2

Question 5

Three, the output of the linear model is not a parameter.

Medium

```
rm(list=ls())
```

Question 1

```
n <-  100
mu <-  rnorm(n, 0, 10)
sigma <-  runif(n, 0, 10)
heights <-  rnorm(n, mu, sigma)
```

Question 2

```
flist <- alist(
  y ~ dnorm(mu, sigma),
  mu ~ dnorm(0,10),
  sigma ~ dunif(0,10)
)
```

Question 3

$$
\begin{aligned}
y_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta x_i \\
\alpha &\sim \text{Normal}(0, 50) \\
\beta &\sim \text{Uniform}(0, 10) \\
\sigma &\sim \text{Uniform}(0, 50)
\end{aligned}
$$

Question 4

$$
\begin{aligned}
h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta y_i \\
\alpha &\sim \text{Normal}(170, 20) \\
\beta &\sim \text{Normal}(0, 2) \\
\sigma &\sim \text{Uniform}(0, 20)
\end{aligned}
$$

Question 3

Yes, it looks like the students are children. If I had known this I would have chosen a diffrent model. Picking such a strong prior was based on adults and that growth even for young students i likely to be zero or close to zero.

Question 4

This depends on your source of information. If you have checked the same data that will be used in the moddeling I might not want to change anything. If this is based on seperate data then it should be taken into account. Assume only 5 students. Having checked that the vairance is small does not mean you should update the prior with this information. That means using the data twice...

Hard

```
rm(list=ls())
```

Question 1

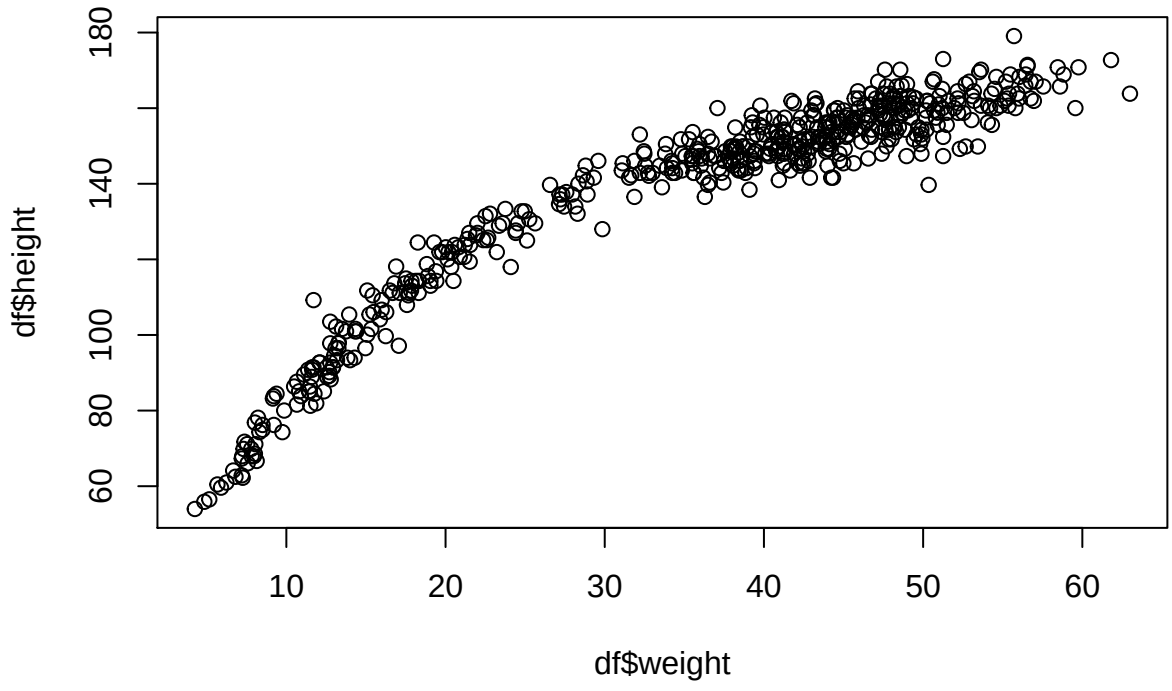
The weights listed below were recorded in the !Kung census, but heights were not recorded for these individuals. Provide predicted heights and 89% intervals (either HPDI or PI) for each of these individuals. That is, fill in the table below, using model-based predictions.

Load and select the data

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

A quick look at the relationship

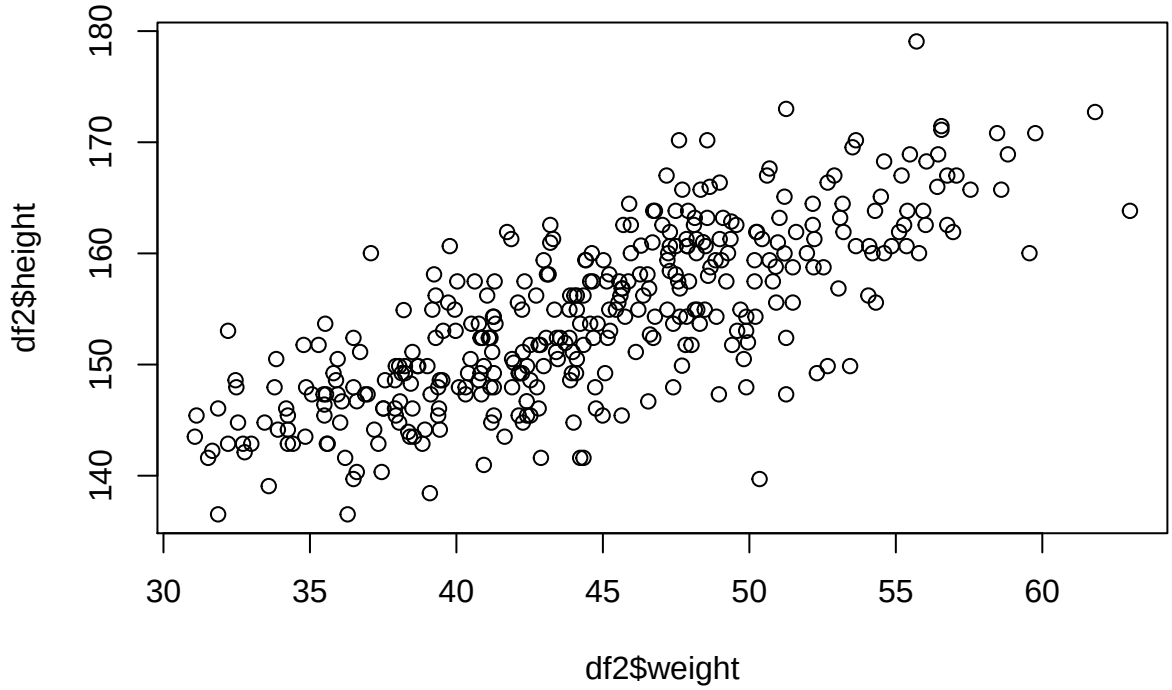
```
plot(df$weight, df$height)
```



The people from the census with missings heights are all above 30 kg. Subsetting the data will make modeling easier beacuse some of the nonlinearity will be taken out. This is good becaues it seems unlikely to be any information about adult height which can be extracted based on data from children. This is true for people with exeptionally low weight also.

```
df2 <-  df[df$weight > 30,]
```

```
plot(df2$weight,df2$height)
```



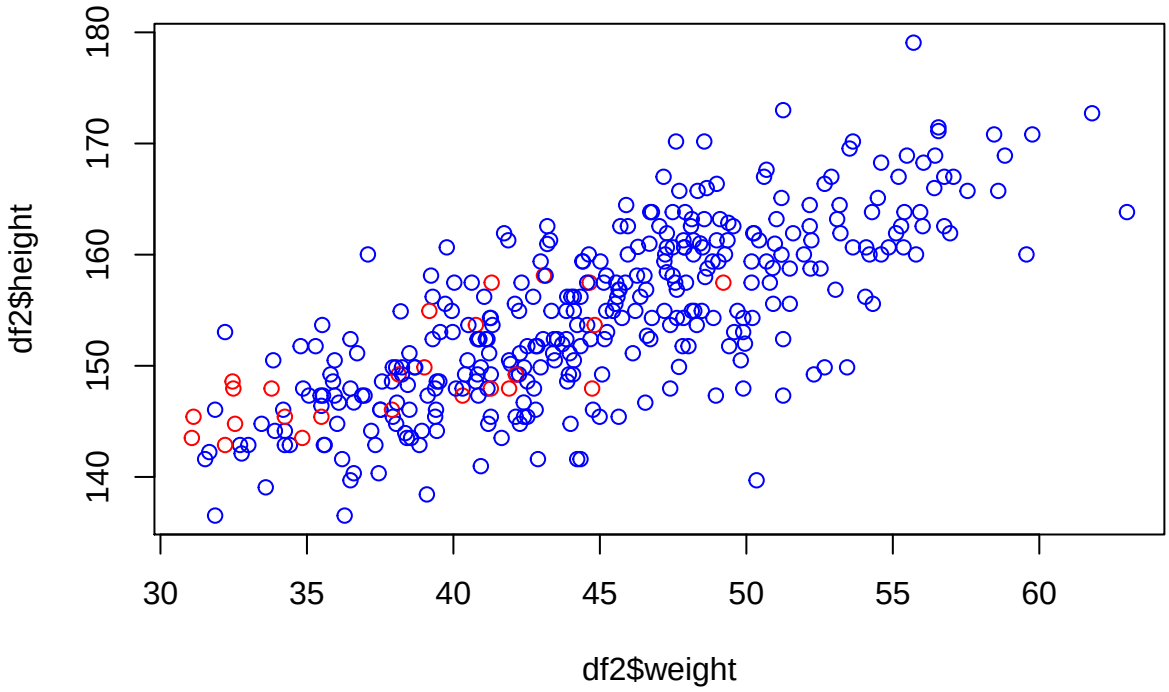
This looks like the same data as the data from only adults. A quick check:

```
sum(df2$age > 18)/nrow(df2)
```

```
## [1] 0.9326146
```

It is close to only the adults in the subset, but it is not guarantee that the census is of adults only.

```
plot(df2$weight,df2$height, col=c("red","blue")[as.numeric(df2$age>18)+1])
```



Since children above 30 kg seems to follow the same patern as adults there is no reason to take the data out. The subset is kept.

Normalizing the weight

```
mean_weight = mean(df2$weight)
sd_weight = sd(df2$weight)
df2$std_weight = (df2$weight-mean_weight)/sd_weight
```

Setting up the model. Adults are normally around 170 but for this same uncertanty is large. Since you are unlikely to increase more than 5 cm per kg it seems like a relativly weak prior to put the variance of b to five. This prior does not include the knowledge that the relationship is most likely positive.

```
flist <- alist(
  height ~ dnorm(mu,sigma),
  mu <- a + b*std_weight,
  a ~ dnorm(170,20),
  b ~ dnorm(0, 5),
  sigma ~ dunif(0,30)
)
```

```
model_q1 <- map(flist, data = df2)
```

Individual weight expected height 89% interval 1

```
table_weights <- c(46.95, 43.72, 64.78, 32.59, 54.63)
table_weights_std <- (table_weights -mean_weight )/sd_weight

height_simulations <- sim(model_q1, data=data.frame(std_weight=table_weights_std))
```

```
## [ 100 / 1000 ]
[ 200 / 1000 ]
[ 300 / 1000 ]
[ 400 / 1000 ]
[ 500 / 1000 ]
```

```
[ 600 / 1000 ]
[ 700 / 1000 ]
[ 800 / 1000 ]
[ 900 / 1000 ]
[ 1000 / 1000 ]
```

```
height_HPDI <- apply( height_simulations, 2, HPDI, prob=0.89)
height_mean <-  apply(height_simulations, 2, mean)
```

```
data.frame(Individual = seq(1,5),
           weight = table_weights,
           mean = height_mean,
           hpdi_low = height_HPDI[1,],
           hpdi_high = height_HPDI[2,] )
```

```
##   Individual weight     mean hpdi_low hpdi_high
## 1          1  46.95 156.5424 148.6186  164.9372
## 2          2  43.72 153.7224 146.6101  162.4311
## 3          3  64.78 172.3505 164.5547  180.4634
## 4          4  32.59 143.3307 135.6264  150.7920
## 5          5  54.63 163.2705 154.9068  171.3735
```

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 map. Present and interpret the estimates. For every 10 units of increase in weight, how much taller does the model predict a child gets?

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

(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.

```
df3 <- df[df$age<18,]
str(df3)
```

```
## 'data.frame':    192 obs. of  4 variables:
##  $ height: num  121.9 105.4 86.4 129.5 109.2 ...
##  $ weight: num  19.6 13.9 10.5 23.6 16 ...
##  $ age   : num  12 8 6.5 13 7 17 16 11 17 8 ...
##  $ male  : int  1 0 0 1 0 1 0 1 0 1 ...
```

Standardising weight makes it easier to reason about the intercept of the linear model. It is the average height. We can guess that that is a 110 cm for children? Another way to look at it is to think about the uncertainty. This depends on age but less than 30cm or more than 190c, should be 3 standard deviations away. In the middle is 110 so that could be the mean. Adding 30 as the standard deviation seams reasonable weak. This gives us the prior for a. **In hindsight this is a linear model, there is no need to scale only to center**

```
sd_weight <- sd(df3$weight)
mean_weight <- mean(df3$weight)
df3$std_weight = (df3$weight - mean_weight) / sd_weight
```

However, reasoning about the prior of the slope is now harder. An infant is about 2 kg (close to zero) and 30 cm. An adult maybe 62 kg and 170. This gives a slope of 140cm/60kg which is aprox 2.3cm/kg. Allowing zero in the one standard deviation intervall means it the prior is quite weak.

Since the weight is standardised both values should be scaled.

```
b_prior_mean = 2.3 * sd_weight
b_prior_sd = 3 * sd_weight
b_prior_mean
```

```
## [1] 20.56042
```

```
b_prior_sd
```

```
## [1] 26.81794
```

```
flist <- alist(
  height ~ dnorm(mu,sigma),
  mu <- a + b*std_weight,
  a ~ dnorm(110,30),
  b ~ dnorm(b_prior_mean, b_prior_sd),
  sigma ~ dunif(0,50)
)
```

```
model_children <- map(flist, data = df3)
```

```
precis(model_children, cor = TRUE)
```

```
##         Mean StdDev   5.5%  94.5% a b sigma
## a     108.32   0.61 107.35 109.29 1 0     0
## b      24.31   0.61  23.34  25.29 0 1     0
## sigma   8.44   0.43   7.75   9.13 0 0     1
```

I happy with the priors. They seem to fit what we see in the data. Plotting mu to have a look at the incline. Because of the weights was scaled

(a) 10 units increase in weights results in about

```
round(24.31 / sd_weight,1) * 10
```

```
## [1] 27
```

27 cm increase in height

```
std_weight_seq <-  seq(-3,3,by=0.05)
mu <-  link(model_children, data = data.frame(std_weight = std_weight_seq))
```

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

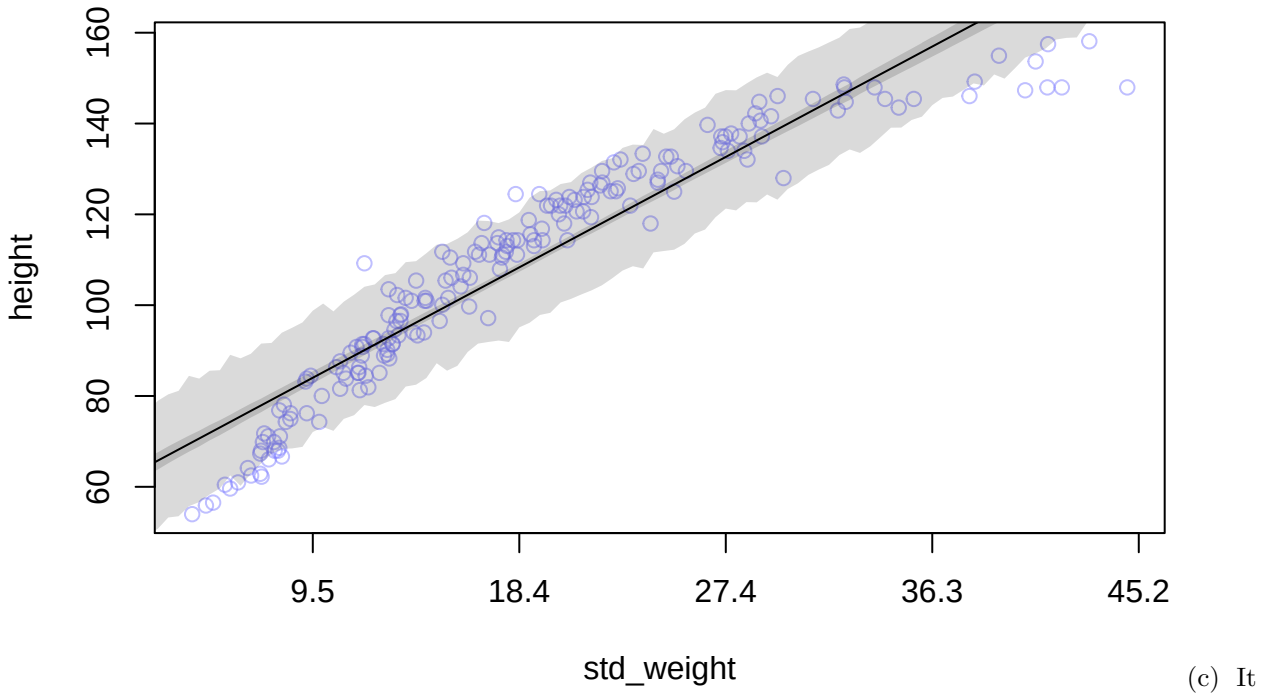
```
height <- sim(model_children, data = data.frame(std_weight = std_weight_seq))
```

```
## [ 100 / 1000 ]
```

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

(b)

```
mu.mean <- apply( mu, 2, mean)
mu.HPDI <- apply( mu, 2, HPDI, prob=0.89)
height.HDPI <-  apply(height, 2, HPDI, prob=0.89)

#plot raw data
#fading out points to make line and interval more visible
plot( height ~ std_weight, data=df3, col=col.alpha(rangi2,0.5), xaxt = "n")
#plot the MAP line, aka the mean mu for each weight
lines(std_weight_seq, mu.mean)
#plot a shaded region for 89% HPDI
shade(mu.HPDI,std_weight_seq)
shade(height.HDPI , std_weight_seq)

at = c(-1, 0, 1, 2, 3)
labels = round(at*sd_weight+mean_weight,1)
axis(1, at =  at , labels = labels)
```



(c) It is not a good model and the uncertainty is irrelevant compared to the poor fit. Taking into account the non-linearity would result in a model which fits the data. This will result in less variance and more accurat predictions.

Question 3

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.

Talking the log does not allow for centering. This means a prior for alpha is harder to get right. I'm going for a standard weak gausian.

```
flist <- alist(
  height ~ dnorm(mu,sigma),
  mu <- alpha + beta*log(weight),
  alpha ~ dnorm(178,100),
  beta ~ dnorm(0,100),
  sigma ~ dunif(0,50)
  )
```

```
log_model <- map(flist, data = df)
```

```
precis(log_model, corr = TRUE)
```

```
##         Mean StdDev   5.5%  94.5% alpha  beta sigma
## alpha -23.78   1.34 -25.92 -21.65  1.00 -0.99     0
## beta   47.08   0.38  46.46  47.69 -0.99  1.00     0
## sigma   5.13   0.16   4.89   5.38  0.00  0.00     1
```

```
weight_seq <-  seq(0,65,by=0.5)
mu <-  link(log_model, data = data.frame(weight = weight_seq))
```

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

```
height <- sim(log_model, data = data.frame(weight = weight_seq))
```

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

```
mu.mean <- apply( mu, 2, mean)
mu.HPDI <- apply( mu, 2, HPDI, prob=0.97)
height.HDPI <-  apply(height, 2, HPDI, prob=0.97)
```

```r
plot(height ~ weight, data = df, col = col.alpha(rangi2, 0.4))
#plot the MAP line, aka the mean mu for each weight
lines(weight_seq, mu.mean)
#plot a shaded region for 89% HPDI
shade(mu.HPDI,weight_seq, col = col.alpha("black", 0.35))
shade(height.HDPI, weight_seq, col = col.alpha("blue", 0.1))
```