Week 5: Bayesian linear regression and introduction to Stan

SHAOHAN CHANG

today

Introduction

Today we will be starting off using Stan, looking at the kid's test score data set (available in resources for the Gelman Hill textbook).

```
library(tidyverse)
library(rstan)
library(tidybayes)
library(here)
```

The data look like this:

```
kidiq <- readRDS("C:/Users/admin/Desktop/Lab 5/kidiq.RDS")
kidiq
```

```
## # A tibble: 434 x 4
##    kid_score mom_hs mom_iq mom_age
##        <int>  <dbl>  <dbl>   <int>
##  1        65      1  121.       27
##  2        98      1   89.4      25
##  3        85      1  115.       27
##  4        83      1   99.4      25
##  5       115      1   92.7      27
##  6        98      0  108.       18
##  7        69      1  139.       20
##  8       106      1  125.       23
##  9       102      1   81.6      24
## 10        95      1   95.1      19
## # ... with 424 more rows
```

As well as the kid's test scores, we have a binary variable indicating whether or not the mother completed high school, the mother's IQ and age.

Descriptives

Question 1

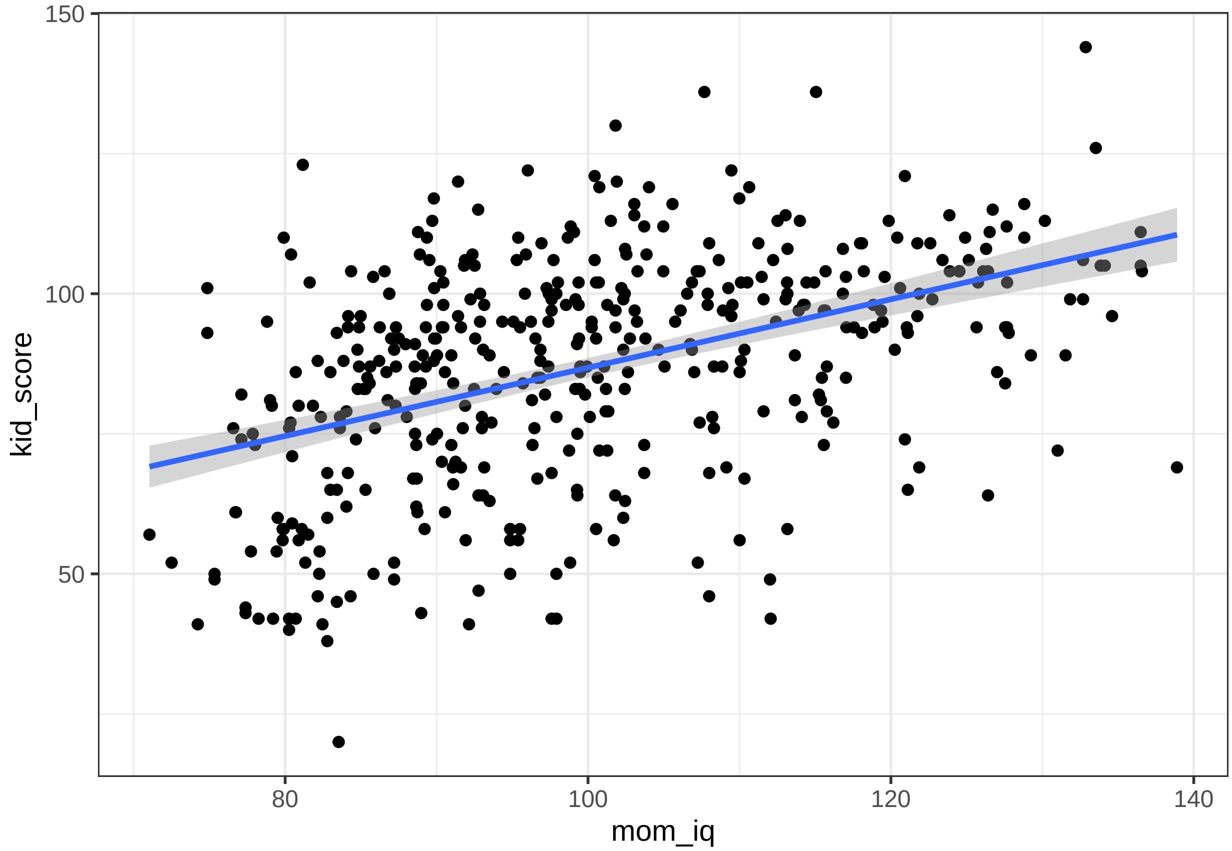
Use plots or tables to show three interesting observations about the data. Remember:

- Explain what your graph/ tables show
- Choose a graph type that's appropriate to the data type

Conclusion by choosing the graph,for the first figure, a scatter plot would be an appropriate graph type to show the relationship between the mother's IQ and the child's score. Each data point represents a mother-child pair, and the x-axis would represent the mother's IQ and the y-axis would represent the child's score. This would allow us to visually observe the trend of increasing child scores with increasing mother's IQ.Compared to the other two figures, it is difficult to find any clear results.

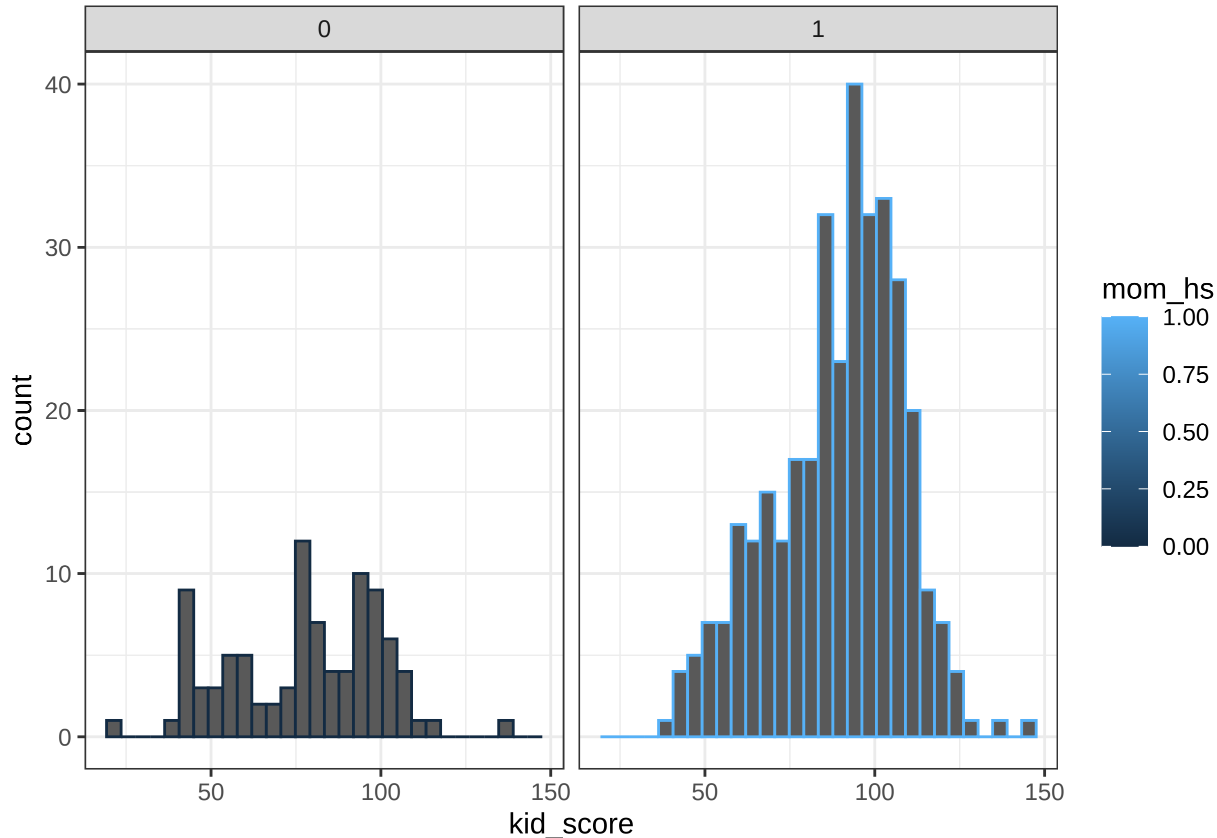
```
library(ggplot2)
library(dplyr)

kidiq %>%ggplot(aes(x = mom_iq, y = kid_score)) + geom_point() +
geom_smooth(method = "lm")+ theme_bw()
```



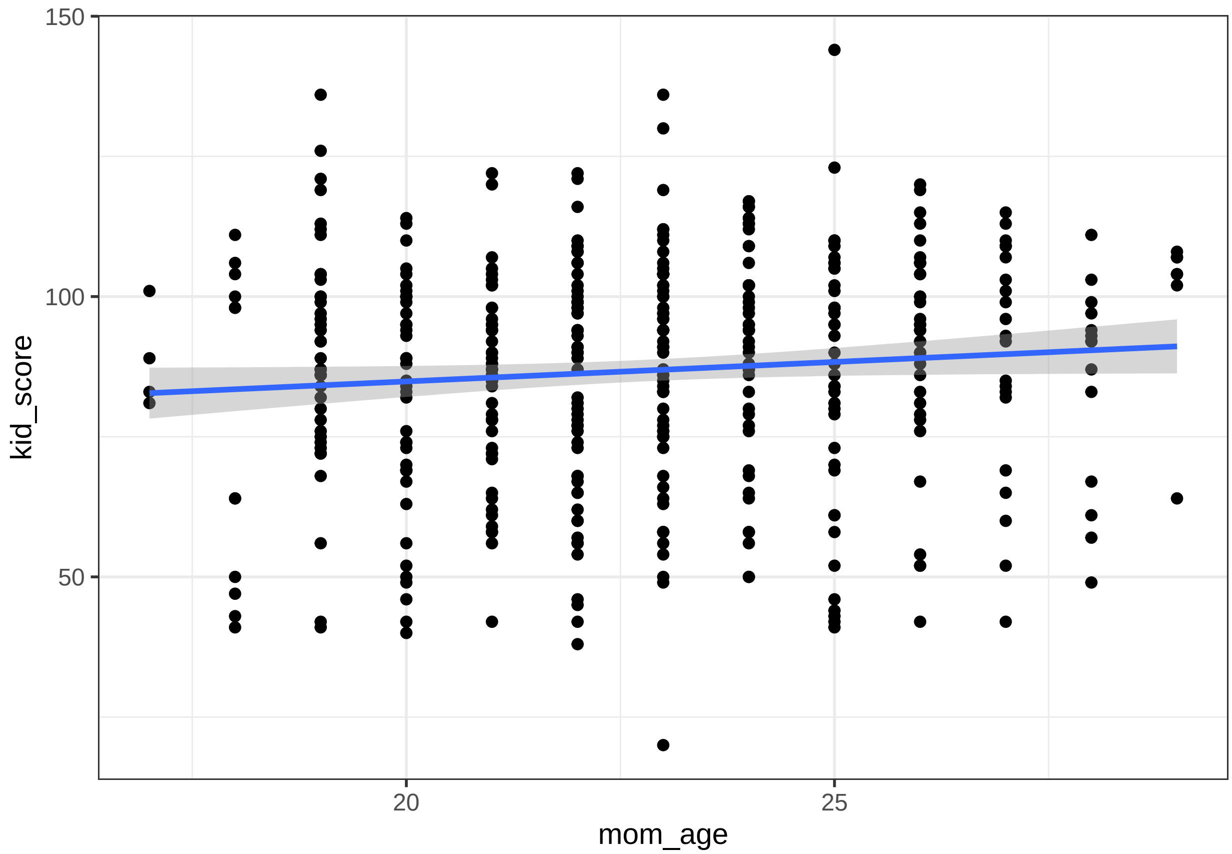
From the above figure, which shows the scores of children tend to increase with their mother's IQ.

```
kidiq %>% group_by(mom_hs)%>%
  ggplot() +
geom_histogram(aes(x = kid_score, colour = mom_hs)) +
theme_bw()+facet_wrap(mom_hs ~ . )
```



The second figure shows the distribution of children's scores is different based on their mother's education level, with those whose mothers did not complete high school having a flatter distribution with less high scores. This is expected as a lack of high school education is often a proxy for lower income and resources. On the other hand, children of mothers who completed high school have higher kid_scores.

```
kidiq %>%ggplot(aes(x = mom_age, y = kid_score)) +geom_point() +
geom_smooth(method = "lm") +theme_bw()
```



The third figure shows that there is minimal correlation between the age of the mother and the score of their child. This suggests that the mother's age may not be a contributing factor to the child's score. The lack of relationship between these two variables is not surprising, but it is also not particularly noteworthy. In summary, the relationship between the mother's age and the child's score is weak and does not appear to be a significant factor. This result is not surprising and does not offer any exceptional insights.

Estimating mean, no covariates

In class we were trying to estimate the mean and standard deviation of the kid's test scores. The `kids2.stan` file contains a Stan model to do this. If you look at it, you will notice the first `data` chunk lists some inputs that we have to define: the outcome variable `y`, number of observations `N`, and the mean and standard deviation of the prior on `mu`. Let's define all these values in a `data` list.

```
y <- kidiq$kid_score
mu0 <- 80
sigma0 <- 10

# named list to input for stan function
data <- list(y = y,
             N = length(y),
             mu0 = mu0,
             sigma0 = sigma0)
```

Now we can run the model:

```
fit <- stan(file ="C:/Users/admin/Desktop/Lab 5/kids2.stan",
            data = data,
```

```r
            chains = 3,
            iter = 500)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.6e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 1: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 1: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 1: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 1: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 1: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 1: Iteration: 251 / 500 [ 50%]  (Sampling)
## Chain 1: Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 1: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 1: Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 1: Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 1: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.003 seconds (Warm-up)
## Chain 1:                0.002 seconds (Sampling)
## Chain 1:                0.005 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 3e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 2: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 2: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 2: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 2: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 2: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 2: Iteration: 251 / 500 [ 50%]  (Sampling)
## Chain 2: Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 2: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 2: Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 2: Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 2: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.004 seconds (Warm-up)
## Chain 2:                0.002 seconds (Sampling)
## Chain 2:                0.006 seconds (Total)
## Chain 2:
##
```

```
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 3: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 3: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 3: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 3: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 3: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 3: Iteration: 251 / 500 [ 50%]  (Sampling)
## Chain 3: Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 3: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 3: Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 3: Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 3: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.005 seconds (Warm-up)
## Chain 3:                0.002 seconds (Sampling)
## Chain 3:                0.007 seconds (Total)
## Chain 3:
```

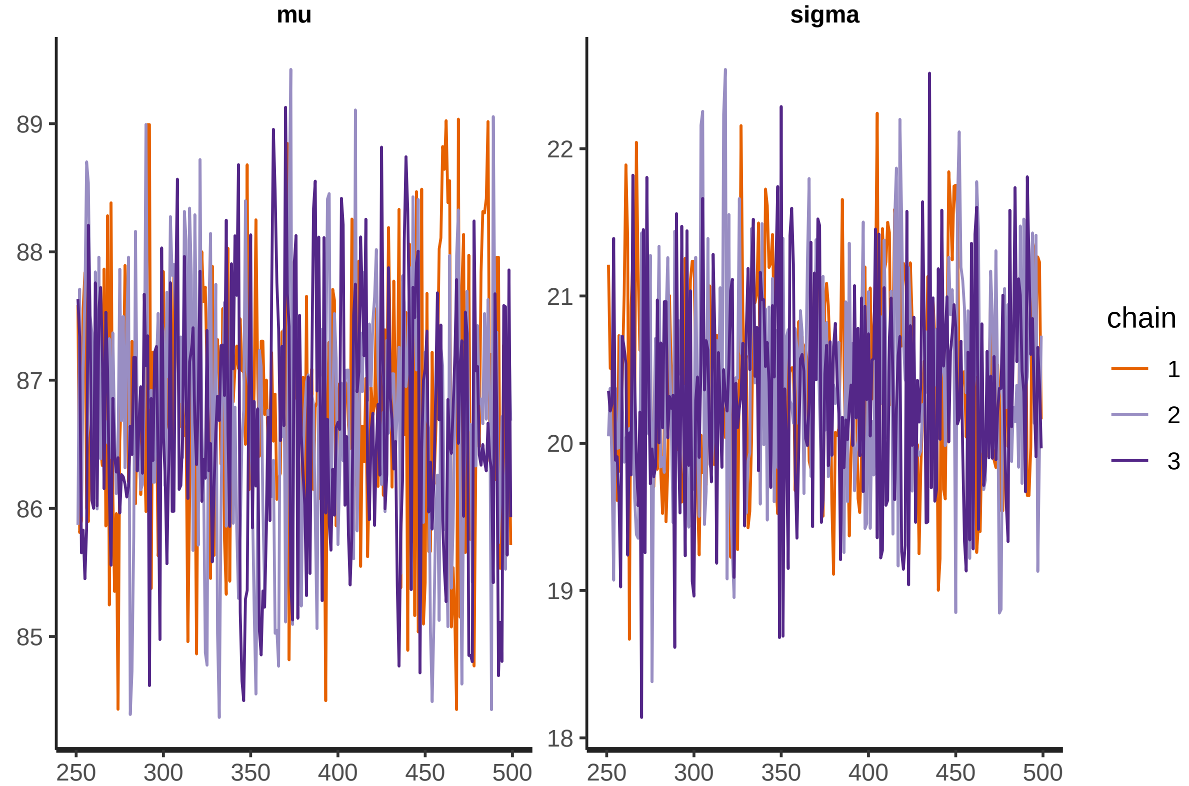
Look at the summary

```
fit
```

```
## Inference for Stan model: anon_model.
## 3 chains, each with iter=500; warmup=250; thin=1;
## post-warmup draws per chain=250, total post-warmup draws=750.
##
##            mean se_mean   sd     2.5%      25%      50%      75%    97.5% n_eff
## mu        86.73    0.04 0.98    84.80    86.07    86.71    87.39    88.65   551
## sigma     20.41    0.03 0.70    19.12    19.92    20.40    20.87    21.78   602
## lp__   -1525.78    0.05 0.97 -1528.22 -1526.26 -1525.46 -1525.06 -1524.79   312
##        Rhat
## mu        1
## sigma     1
## lp__      1
##
## Samples were drawn using NUTS(diag_e) at Sun Feb 12 13:04:00 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

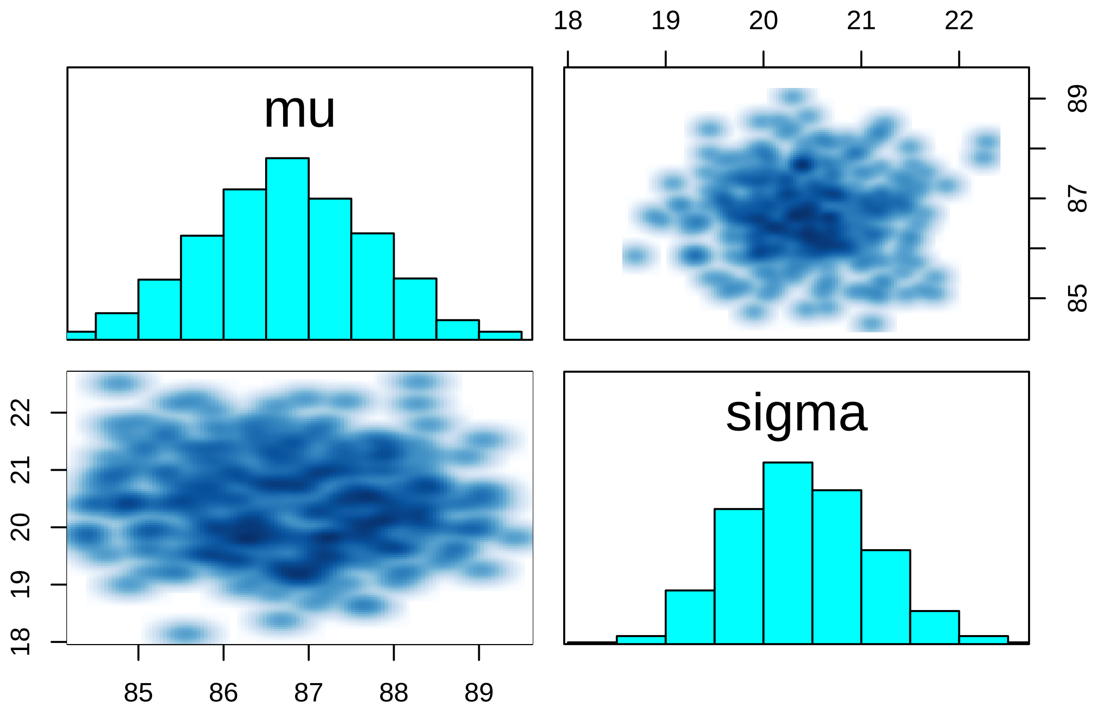
Traceplot

```
traceplot(fit)
```

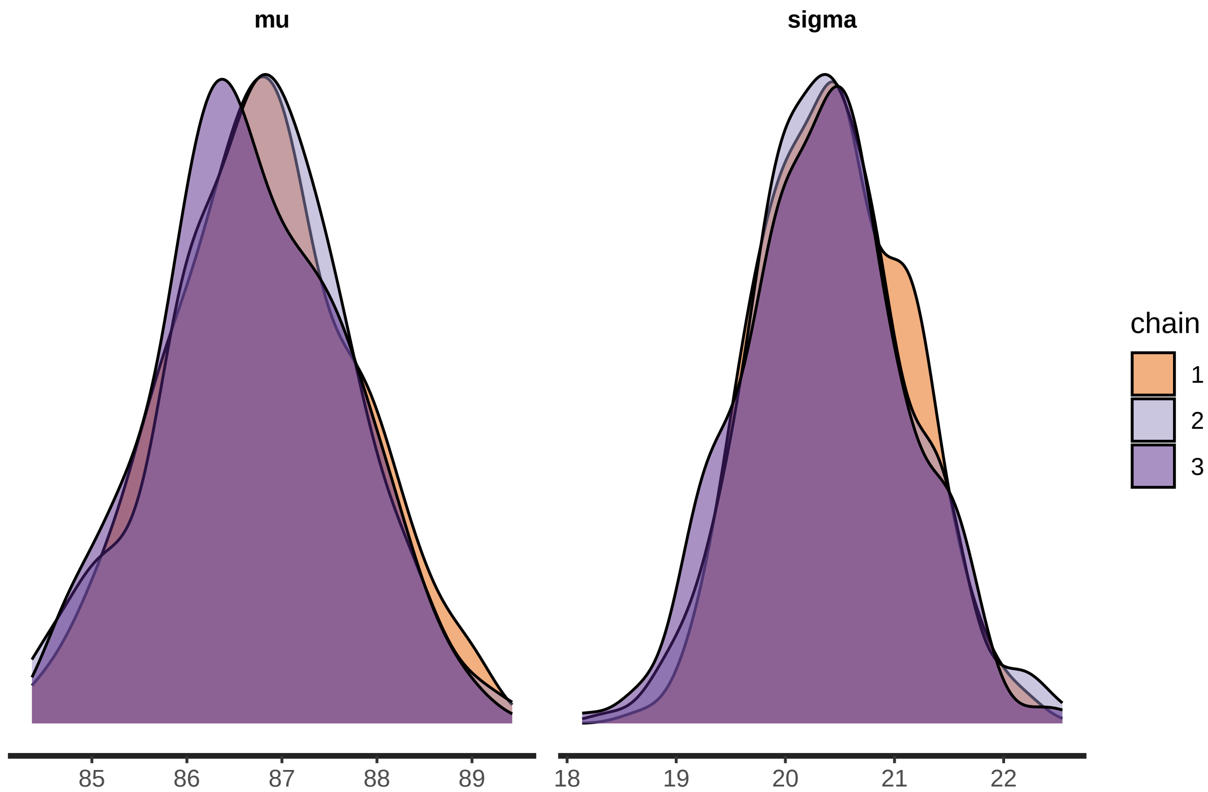


All looks fine.

```
pairs(fit, pars = c("mu", "sigma"))
```




```
stan_dens(fit, separate_chains = TRUE)
```



Understanding output

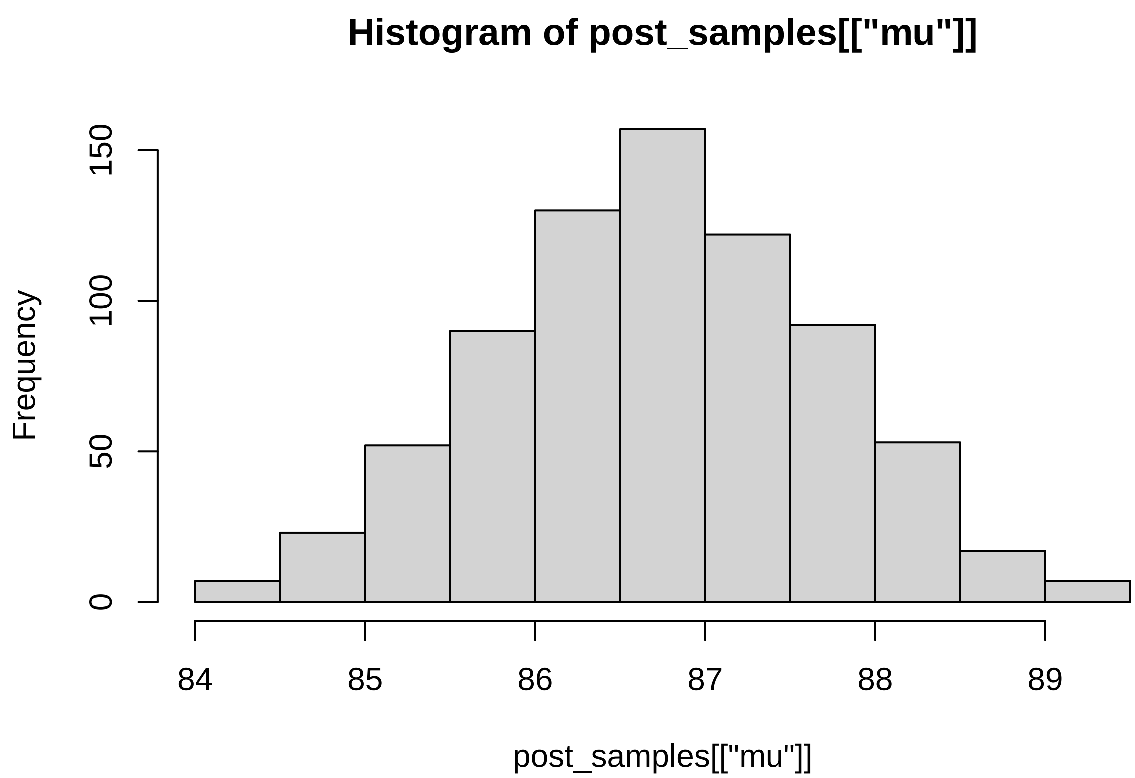
What does the model actually give us? A number of samples from the posteriors. To see this, we can use `extract` to get the samples.

```
post_samples <- extract(fit)
head(post_samples[["mu"]])
```

```
## [1] 86.83101 85.75347 89.01624 85.24656 87.09402 88.82176
```

This is a list, and in this case, each element of the list has 4000 samples. E.g. quickly plot a histogram of mu

```
hist(post_samples[["mu"]])
```




```
median(post_samples[["mu"]])
```

```
## [1] 86.708
```

```
# 95% bayesian credible interval
quantile(post_samples[["mu"]], 0.025)
```

```
##     2.5%
## 84.79682
```

```
quantile(post_samples[["mu"]], 0.975)
```

```
##    97.5%
## 88.65375
```

Plot estimates

There are a bunch of packages, built-in functions that let you plot the estimates from the model, and I encourage you to explore these options (particularly in `bayesplot`, which we will most likely be using later on). I like using the `tidybayes` package, which allows us to easily get the posterior samples in a tidy format (e.g. using gather draws to get in long format). Once we have that, it's easy to just pipe and do ggplots as usual.

Get the posterior samples for mu and sigma in long format:

```
draw_samples <- fit  %>%
  gather_draws(mu, sigma) # gather = long format

head(draw_samples)
```

```
## # A tibble: 6 x 5
## # Groups:   .variable [1]
##   .chain .iteration .draw .variable .value
##    <int>      <int> <int> <chr>      <dbl>
## 1      1          1     1 mu          87.6
## 2      1          2     2 mu          85.8
## 3      1          3     3 mu          85.9
## 4      1          4     4 mu          87.6
## 5      1          5     5 mu          87.8
## 6      1          6     6 mu          86.0
```

```
# wide format
fit  %>%  spread_draws(mu, sigma)
```

```
## # A tibble: 750 x 5
##    .chain .iteration .draw    mu sigma
##     <int>      <int> <int> <dbl> <dbl>
##  1      1          1     1  87.6  21.2
##  2      1          2     2  85.8  20.5
##  3      1          3     3  85.9  20.5
##  4      1          4     4  87.6  19.8
##  5      1          5     5  87.8  20.0
##  6      1          6     6  86.0  19.6
##  7      1          7     7  85.9  20.7
##  8      1          8     8  86.6  19.8
##  9      1          9     9  86.8  20.5
## 10      1         10    10  87.0  21.0
## # ... with 740 more rows
```

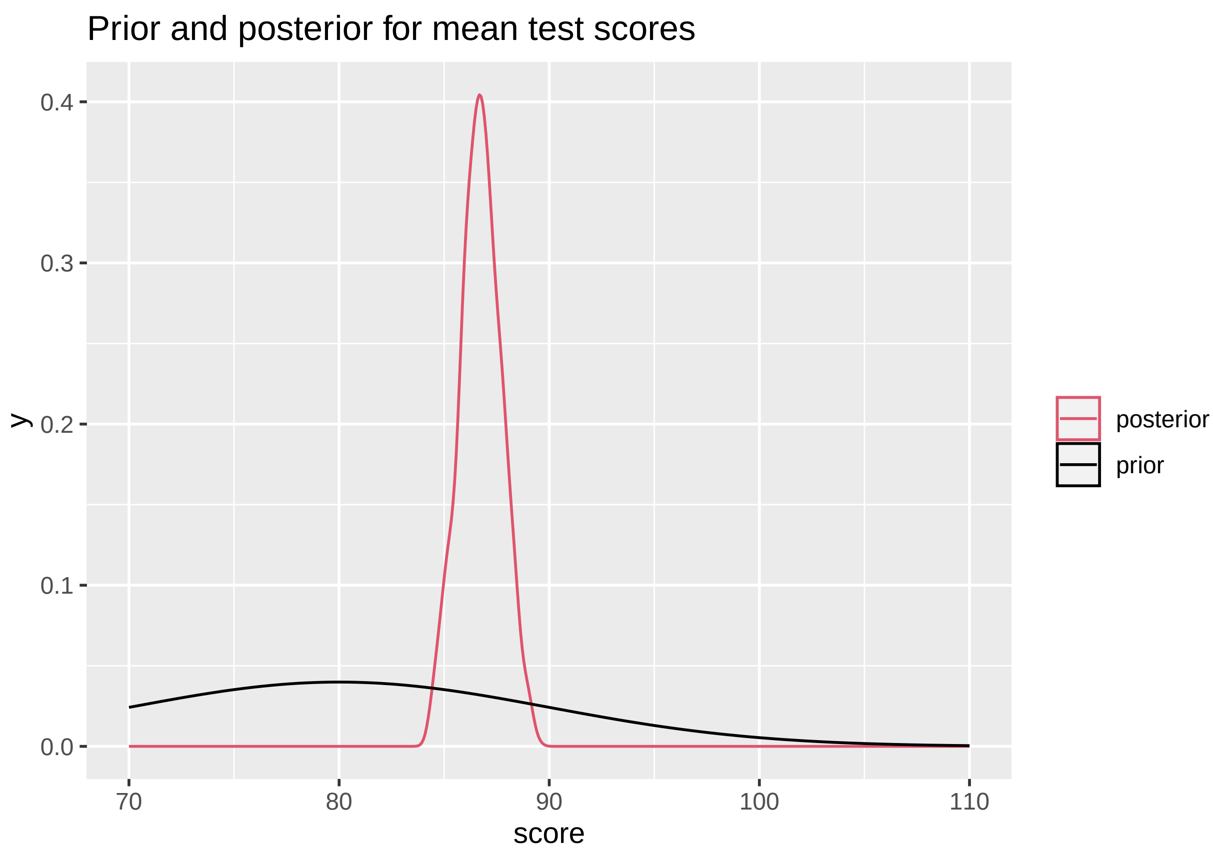
```
# quickly calculate the quantiles using

draw_samples %>% median_qi(.width = 0.8)
```

```
## # A tibble: 2 x 7
##   .variable .value .lower .upper .width .point .interval
##   <chr>      <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
## 1 mu          86.7   85.4   88.0    0.8 median qi
## 2 sigma       20.4   19.5   21.4    0.8 median qi
```

Let's plot the density of the posterior samples for mu and add in the prior distribution

```
draw_samples %>%
  filter(.variable == "mu") %>%
  ggplot(aes(.value, color = "posterior")) + geom_density() +
  xlim(c(70, 110)) +
  stat_function(fun = dnorm,
        args = list(mean = mu0,
                    sd = sigma0),
        aes(colour = 'prior')) +
  scale_color_manual(name = "", values = c("prior" = 1, "posterior" = 2)) +
  xlab("score")+ggtitle("Prior and posterior for mean test scores")
```



Question 2

Change the prior to be much more informative (by changing the standard deviation to be 0.1). Rerun the model. Do the estimates change? Plot the prior and posterior densities.

```
sigma0 = 0.1 # Value setting
data2 <- list(y = y,N = length(y),mu0 = mu0,sigma0 = sigma0)
fit2 <- stan(file ="C:/Users/admin/Desktop/Lab 5/kids2.stan",data = data2) # The file of kids2.stan
```

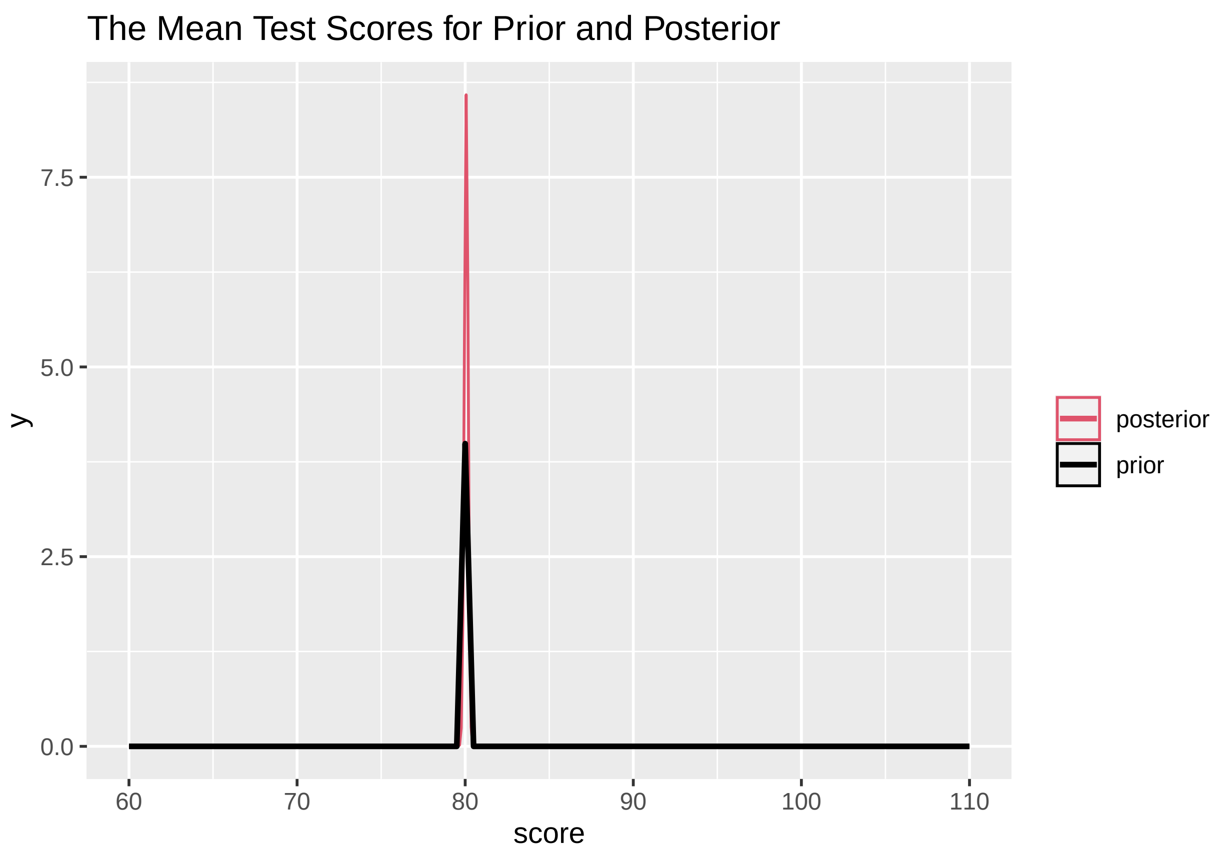
```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 5e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

```
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.009 seconds (Warm-up)
## Chain 1:                0.009 seconds (Sampling)
## Chain 1:                0.018 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 3e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.009 seconds (Warm-up)
## Chain 2:                0.008 seconds (Sampling)
## Chain 2:                0.017 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 7e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.01 seconds (Warm-up)
```

```
## Chain 3:                0.009 seconds (Sampling)
## Chain 3:                0.019 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.009 seconds (Warm-up)
## Chain 4:                0.008 seconds (Sampling)
## Chain 4:                0.017 seconds (Total)
## Chain 4:
```

```
draw_samples2<- fit2 %>% gather_draws(mu, sigma)

draw_samples2%>%filter(.variable == "mu") %>%
  ggplot(aes(.value, color = "posterior")) +
  geom_density() +xlim(c(60, 110))+xlab("score") +
  stat_function(fun = dnorm,args = list(mean = mu0,sd = sigma0),aes(colour = 'prior'), size = 1) +
  scale_color_manual(name = "", values = c("prior" = 1, "posterior" = 2)) +
  ggtitle("The Mean Test Scores for Prior and Posterior")
```



Adding covariates

Now let's see how kid's test scores are related to mother's education. We want to run the simple linear regression

$$Score = \alpha + \beta X$$

where $X = 1$ if the mother finished high school and zero otherwise.

`kid3.stan` has the stan model to do this. Notice now we have some inputs related to the design matrix X and the number of covariates (in this case, it's just 1).

Let's get the data we need and run the model.

```
X <- as.matrix(kidiq$mom_hs, ncol = 1) # force this to be a matrix
K <- 1
data <- list(y = y, N = length(y),
             X =X, K = K)
fit2 <- stan(file = here("C:/Users/admin/Desktop/Lab 5/kids3.stan"), # fit2 represent the kids3 file
            data = data,
            iter = 1000)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 6e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.6 seconds.
```

```
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.068 seconds (Warm-up)
## Chain 1:                0.032 seconds (Sampling)
## Chain 1:                0.1 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 8e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.054 seconds (Warm-up)
## Chain 2:                0.033 seconds (Sampling)
## Chain 2:                0.087 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:   1 / 1000 [  0%]  (Warmup)
```

```
## Chain 3: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.069 seconds (Warm-up)
## Chain 3:                0.033 seconds (Sampling)
## Chain 3:                0.102 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 8e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.056 seconds (Warm-up)
## Chain 4:                0.035 seconds (Sampling)
## Chain 4:                0.091 seconds (Total)
## Chain 4:
```

Question 3

a) Confirm that the estimates of the intercept and slope are comparable to results from `lm()`

```r
lm_model<- lm(kid_score ~ mom_hs,data=kidiq)
summary(lm_model)
```

```
##
## Call:
## lm(formula = kid_score ~ mom_hs, data = kidiq)
##
## Residuals:
##    Min     1Q Median     3Q    Max
```

```
## -57.55 -13.32   2.68  14.68  58.45
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   77.548      2.059  37.670  < 2e-16 ***
## mom_hs        11.771      2.322   5.069 5.96e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.85 on 432 degrees of freedom
## Multiple R-squared:  0.05613,    Adjusted R-squared:  0.05394
## F-statistic: 25.69 on 1 and 432 DF,  p-value: 5.957e-07
```

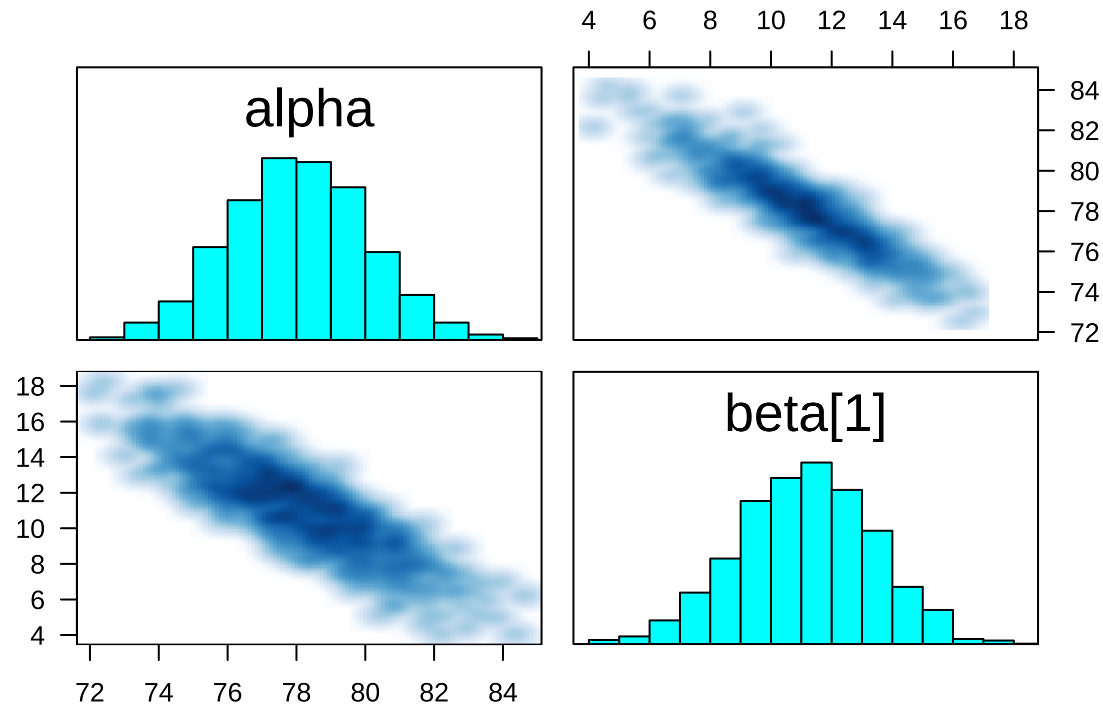
```
fit2
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##             mean se_mean   sd     2.5%      25%      50%      75%    97.5%
## alpha      78.06    0.07 2.00    74.11    76.64    78.06    79.41    81.97
## beta[1]    11.14    0.07 2.23     6.73     9.66    11.22    12.66    15.39
## sigma      19.82    0.02 0.65    18.60    19.37    19.78    20.26    21.16
## lp__    -1514.32    0.04 1.17 -1517.46 -1514.88 -1514.02 -1513.47 -1512.99
##         n_eff Rhat
## alpha     916    1
## beta[1]   885    1
## sigma    1126    1
## lp__      886    1
##
## Samples were drawn using NUTS(diag_e) at Sun Feb 12 13:04:27 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

The results of two models, one fitted by Stan and one fitted by a linear model, were compared for the estimates of the intercept and slope. The Stan model estimated the intercept to be 78 and the slope to be 11, while the linear model estimated the intercept to be 77.5 and the slope to be 11. The results of these two models were found to be similar.

b) Do a `pairs` plot to investigate the joint sample distributions of the slope and intercept. Comment briefly on what you see. Is this potentially a problem?

```
pairs(fit2,pars=c("alpha","beta"),las = 1)
```

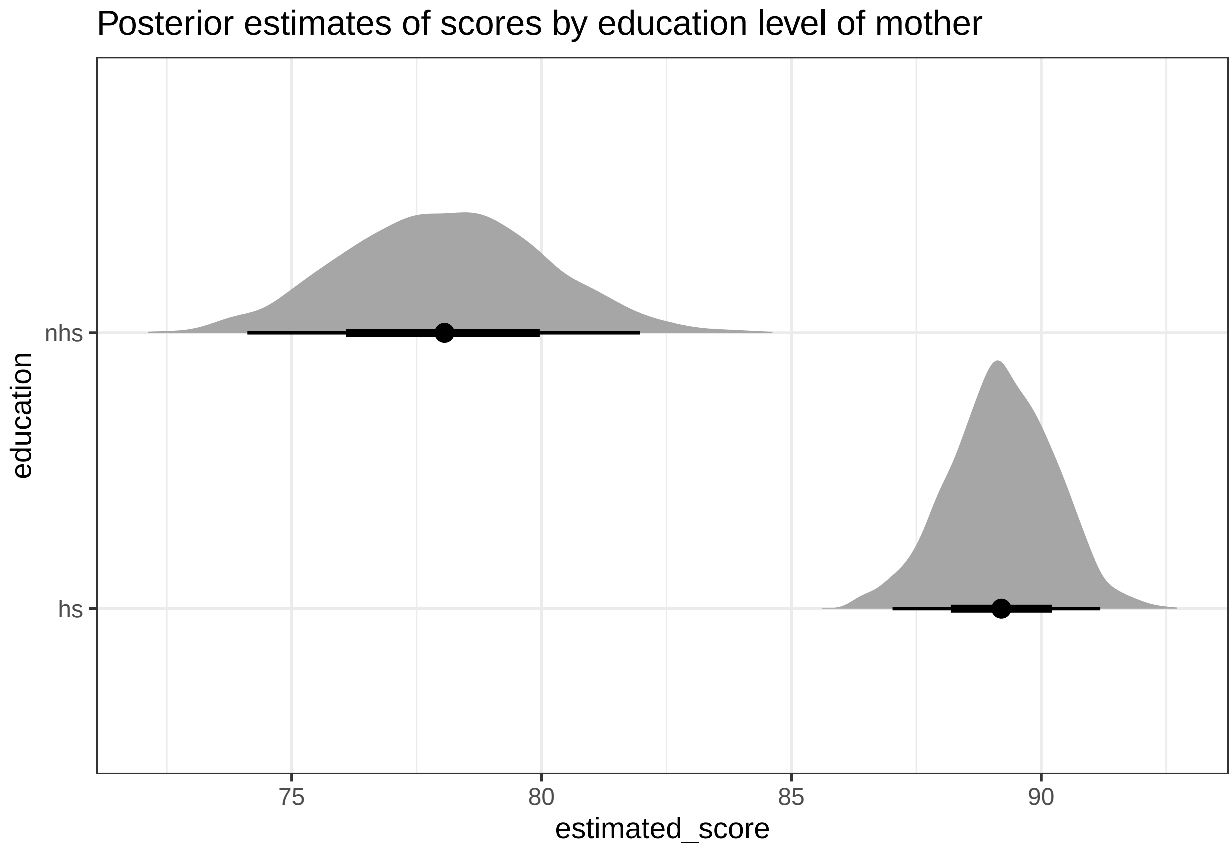


The joint distribution of the slope and intercept estimations was analyzed, and it was found that there may be other factors affecting the child's score besides the mother's education level. This is because the distribution of the intercept estimates showed a relatively large degree of variability.

Plotting results

It might be nice to plot the posterior samples of the estimates for the non-high-school and high-school mothered kids. Here's some code that does this: notice the `beta[condition]` syntax. Also notice I'm using `spread_draws`, because it's easier to calculate the estimated effects in wide format

```
fit2 |>
  spread_draws(alpha, beta[k], sigma) |>
     mutate(nhs = alpha, # no high school is just the intercept
          hs = alpha + beta) |>
  select(nhs, hs) |>
  pivot_longer(nhs:hs, names_to = "education",
               values_to = "estimated_score") |>
  ggplot(aes(y = education, x = estimated_score)) +
  stat_halfeye() +
  theme_bw() +
  ggtitle("Posterior estimates of scores by education level of mother")
```



Question 4

Add in mother's IQ as a covariate and rerun the model. Please mean center the covariate before putting it into the model. Interpret the coefficient on the (centered) mum's IQ.

```
kidiq <- kidiq %>%
mutate(centered_iq = scale(mom_iq, scale = FALSE))
X <- cbind(as.matrix(kidiq$mom_hs, ncol = 1), as.matrix(kidiq$centered_iq, ncol = 1))

data4 <- list(y = y, N = length(y),X = X, K = 2)
fit4 <- stan(file="C:/Users/admin/Desktop/Lab 5/kids3.stan",data = data4,iter = 500)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.17 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 1: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 1: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 1: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 1: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 1: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 1: Iteration: 251 / 500 [ 50%]  (Sampling)
```

```
## Chain 1: Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 1: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 1: Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 1: Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 1: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.048 seconds (Warm-up)
## Chain 1:                0.018 seconds (Sampling)
## Chain 1:                0.066 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 9e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 2: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 2: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 2: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 2: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 2: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 2: Iteration: 251 / 500 [ 50%]  (Sampling)
## Chain 2: Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 2: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 2: Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 2: Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 2: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.05 seconds (Warm-up)
## Chain 2:                0.021 seconds (Sampling)
## Chain 2:                0.071 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 8e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 3: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 3: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 3: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 3: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 3: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 3: Iteration: 251 / 500 [ 50%]  (Sampling)
## Chain 3: Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 3: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 3: Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 3: Iteration: 450 / 500 [ 90%]  (Sampling)
```

```
## Chain 3: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.046 seconds (Warm-up)
## Chain 3:                0.018 seconds (Sampling)
## Chain 3:                0.064 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 4: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 4: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 4: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 4: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 4: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 4: Iteration: 251 / 500 [ 50%]  (Sampling)
## Chain 4: Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 4: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 4: Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 4: Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 4: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.048 seconds (Warm-up)
## Chain 4:                0.023 seconds (Sampling)
## Chain 4:                0.071 seconds (Total)
## Chain 4:
```

```
fit4
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=500; warmup=250; thin=1;
## post-warmup draws per chain=250, total post-warmup draws=1000.
##
##             mean se_mean   sd     2.5%      25%      50%      75%    97.5%
## alpha      82.35    0.08 1.89    78.64    81.08    82.39    83.63    85.99
## beta[1]     5.69    0.09 2.11     1.55     4.24     5.65     7.07     9.99
## beta[2]     0.57    0.00 0.06     0.46     0.53     0.57     0.61     0.69
## sigma      18.13    0.03 0.62    16.95    17.73    18.11    18.55    19.39
## lp__    -1474.43    0.07 1.41 -1478.06 -1475.15 -1474.12 -1473.39 -1472.63
##         n_eff Rhat
## alpha     497    1
## beta[1]   522    1
## beta[2]   615    1
## sigma     566    1
## lp__      469    1
##
## Samples were drawn using NUTS(diag_e) at Sun Feb 12 13:04:27 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Interpretation: For every one point increase above average in the mother's IQ, the estimated score of the child is expected to rise by approximately 0.57, according to the estimate of the coefficient.

Question 5

Confirm the results from Stan agree with `lm()`

```
lm_model2<- lm(kid_score ~ mom_hs + mom_iq,data = kidiq)
summary(lm_model2)
```

```
##
## Call:
## lm(formula = kid_score ~ mom_hs + mom_iq, data = kidiq)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.873 -12.663   2.404  11.356  49.545
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 25.73154    5.87521   4.380 1.49e-05 ***
## mom_hs       5.95012    2.21181   2.690  0.00742 **
## mom_iq       0.56391    0.06057   9.309  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.14 on 431 degrees of freedom
## Multiple R-squared:  0.2141, Adjusted R-squared:  0.2105
## F-statistic: 58.72 on 2 and 431 DF,  p-value: < 2.2e-16
```

The coefficient estimate for the mother's IQ in the linear model (LM) was approximately 0.56, which is consistent with the estimate from the Stan model. The estimate for the mother's education level (mum_hs) in LM was about 5.95, which is similar to the estimate from Stan. However, the estimate of the intercept in LM was 25.7, whereas the estimate from Stan was 82.39.

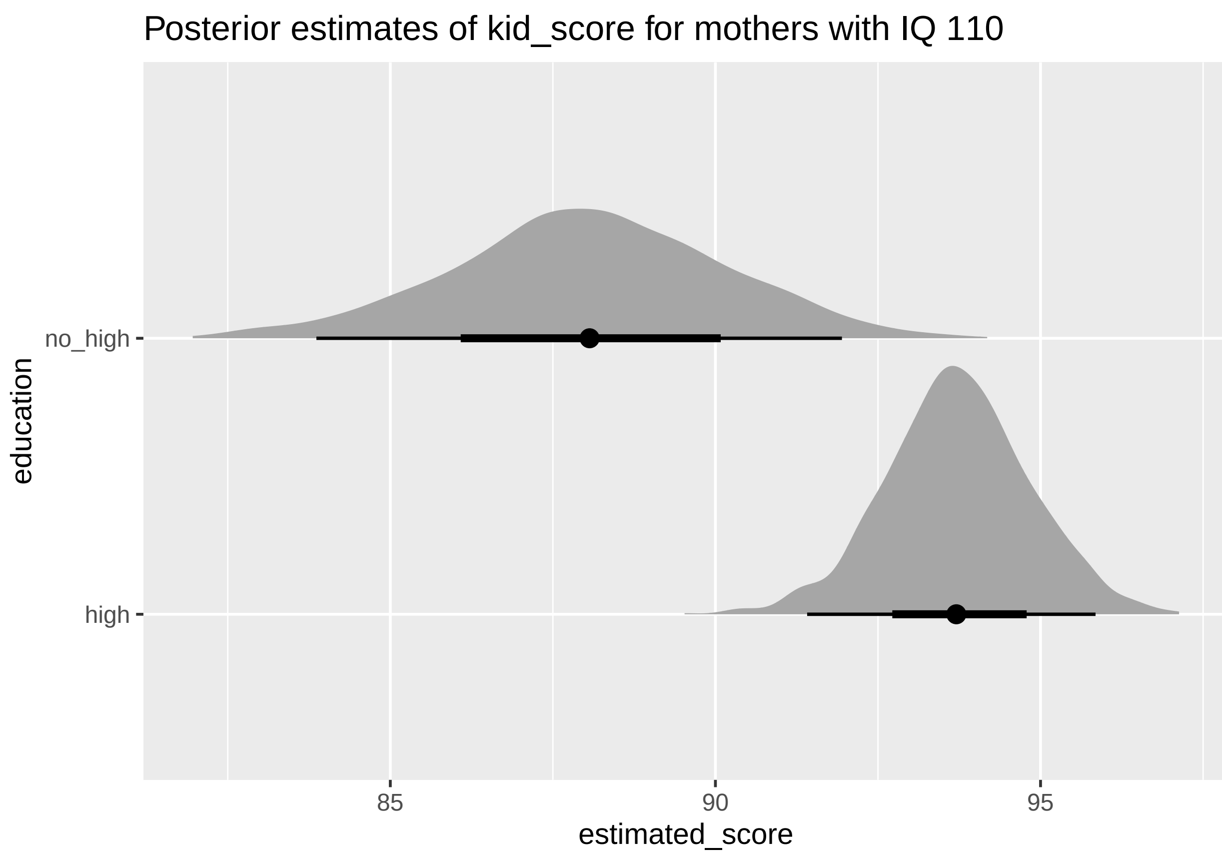
Question 6

Plot the posterior estimates of scores by education of mother for mothers who have an IQ of 110.

```
mean(kidiq$mom_iq)
```

```
## [1] 100
```

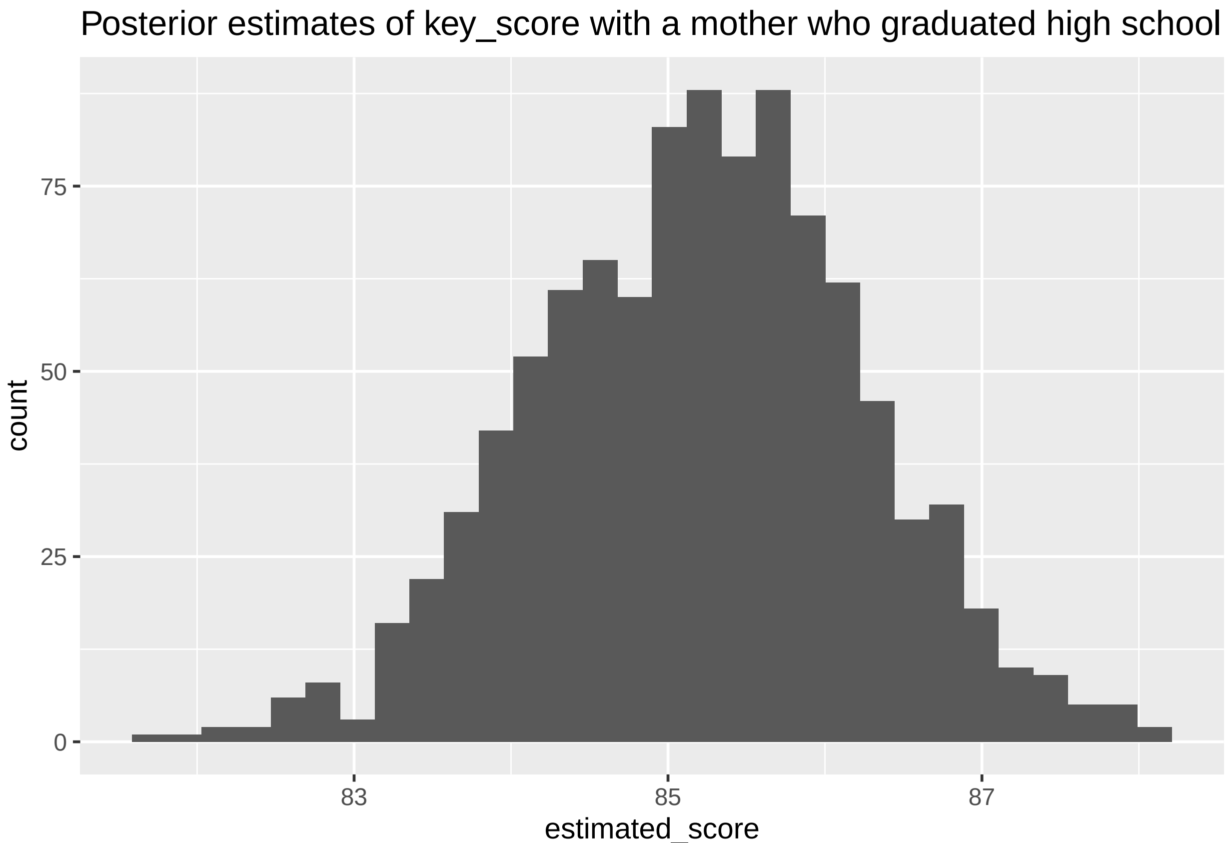
```
fit4 %>%spread_draws(alpha, beta[condition], sigma) %>%
pivot_wider(names_from = condition, names_prefix = "beta", values_from = beta) %>%
transmute(no_high = alpha + beta2 * 10, high = alpha + beta1 + beta2 * 10) %>%
pivot_longer(cols = c(no_high, high), names_to = "education", values_to = "estimated_score") %>%
ggplot(aes(y = education, x = estimated_score)) + stat_halfeye() +
ggtitle("Posterior estimates of kid_score for mothers with IQ 110")
```



Question 7

Generate and plot (as a histogram) samples from the posterior predictive distribution for a new kid with a mother who graduated high school and has an IQ of 95.

```
fit4 %>%spread_draws(alpha, beta[condition], sigma) %>%
pivot_wider(names_from = condition, names_prefix = "beta", values_from = beta) %>%
mutate(hs = alpha + beta1 + beta2 *(-5)) %>%
pivot_longer(hs, names_to = "education", values_to = "estimated_score") %>%
ggplot(aes( x = estimated_score)) +geom_histogram() +
ggtitle("Posterior estimates of key_score with a mother who graduated high school and has an IQ of 95")
```


Posterior estimates of key_score with a mother who graduated high school a
count
75
50
25
0
83
85
87
estimated_score