Lab 5

Roxanne Li

2024-02-07

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

Reading in the data:

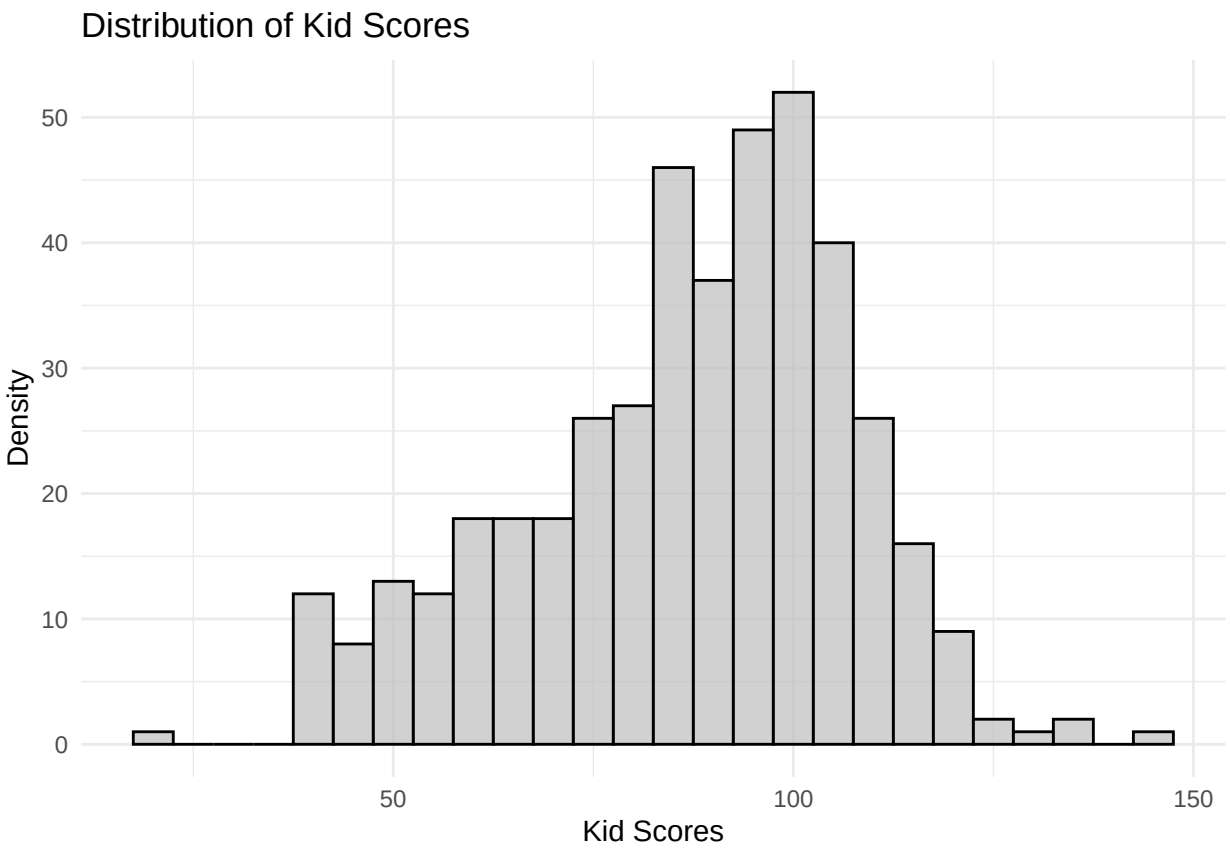
```
kid <- read_rds("data/Kidiq.RDS")
head(kid)
```

```
## # A tibble: 6 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
```

Q1

Plot the distribution of kid scores:

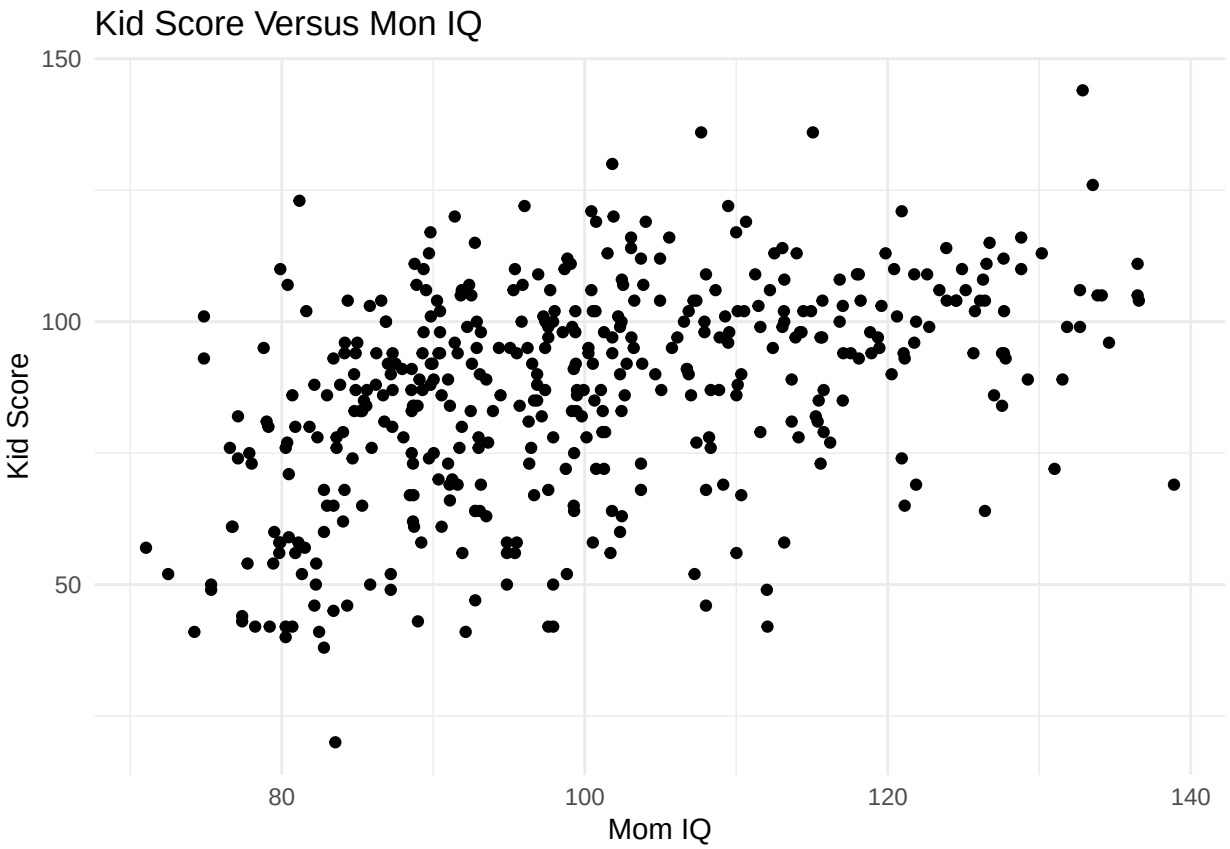
```
kid |>
  ggplot(aes(kid_score)) +
  geom_histogram(binwidth = 5, fill = "grey", color = "black", alpha = 0.7) +
  labs(title = "Distribution of Kid Scores", x = "Kid Scores", y = "Density") +
  theme_minimal()
```



The histogram shows that the kid scores are distributed almost normally between 0 and 150.

Plotting kid score versus mom IQ:

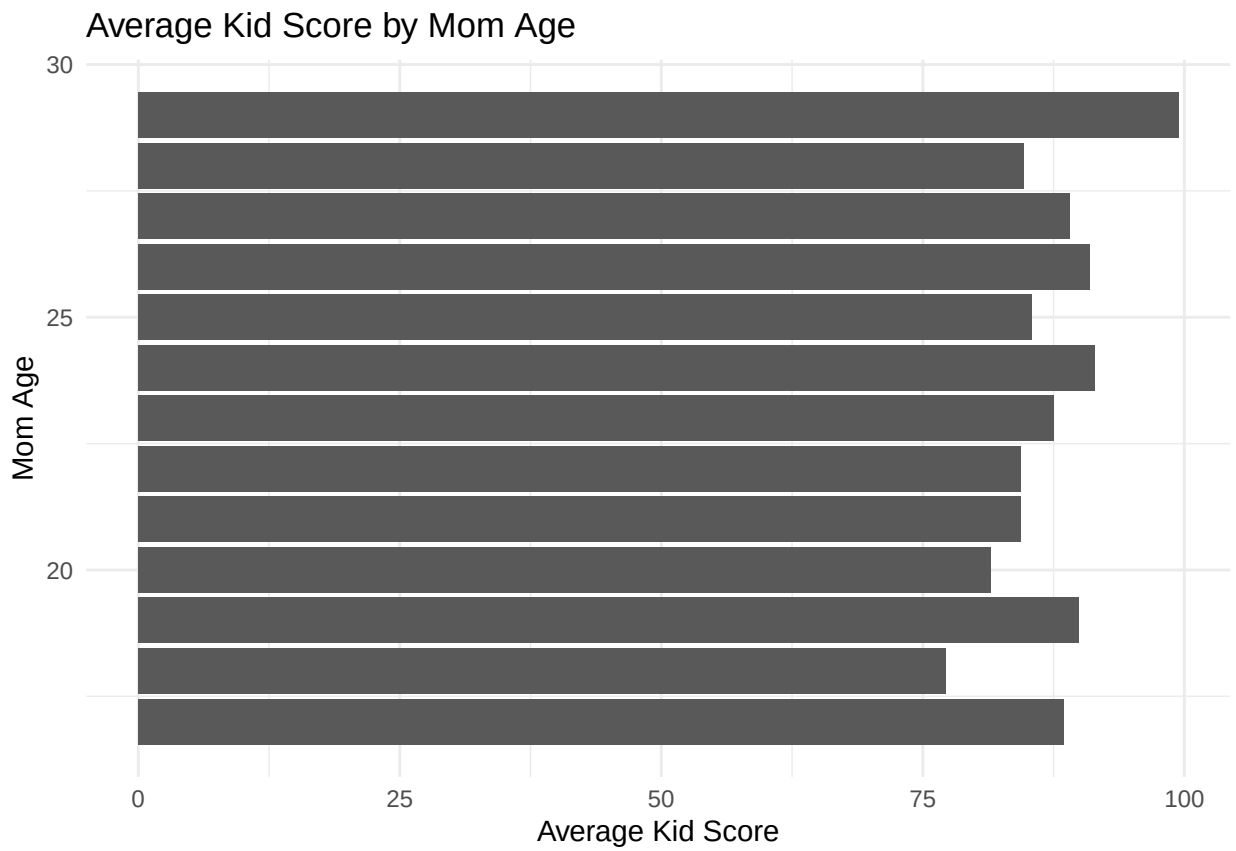
```
kid |>
  ggplot(aes(mom_iq, kid_score)) +
  geom_point() +
  labs(title = "Kid Score Versus Mon IQ", x = "Mom IQ", y = "Kid Score") +
  theme_minimal()
```



The scatter plot shows a positive linear relation between kid scores and mom IQ's.

Plotting kid score versus mom age:

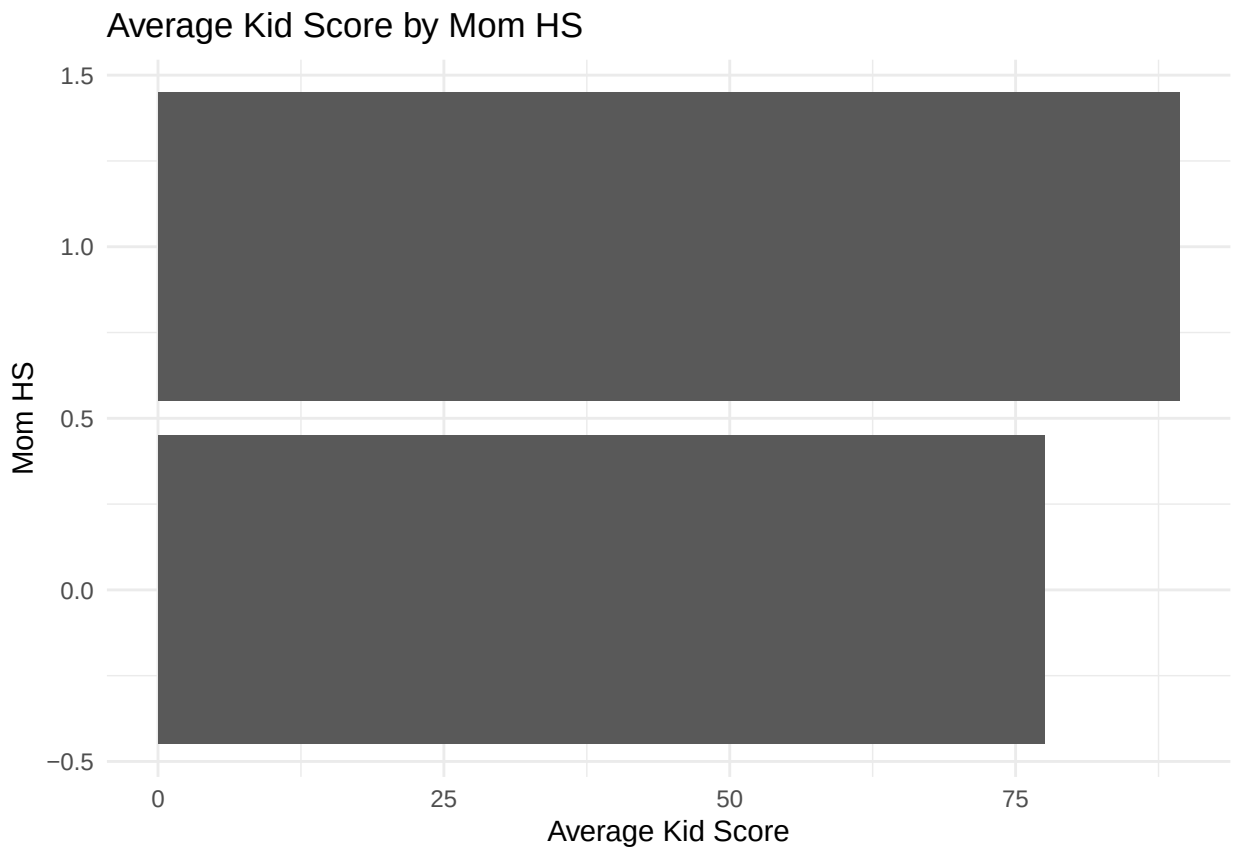
```
kid |>
  group_by(mom_age) |>
  summarise(avg_kid_score = mean(kid_score)) |>
  ggplot(aes(x=avg_kid_score, y=mom_age)) +
  geom_bar(stat = "identity", orientation = 'y') +
  labs(title = "Average Kid Score by Mom Age",
       x = "Average Kid Score", y = "Mom Age") +
  theme_minimal()
```



The bar plot of average kid score by mom ages show that when mom ages are between 20 and 25, the average kid scores are relatively lower than when mom ages are above 25 or below 20.

Plotting kid score versus mom high school status:

```
kid |>
  group_by(mom_hs) |>
  summarise(avg_kid_score = mean(kid_score)) |>
  ggplot(aes(x=avg_kid_score, y=mom_hs)) +
  geom_bar(stat = "identity", orientation = 'y') +
  labs(title = "Average Kid Score by Mom HS",
       x = "Average Kid Score", y = "Mom HS") +
  theme_minimal()
```



The bar plot shows that the average kid score of moms who went to high school is higher than moms who didn't attend high school.

Q2

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.

```
y <- kid$kid_score
mu0 <- 80
sigma0 <- 0.1
# named list to input for stan function
data1 <- list(y = y,
              N = length(y),
              mu0 = mu0,
              sigma0 = sigma0)
```

Run the model:

```
fit1 <- stan(file = "code/models/kids2.stan",
               data = data1,
               chains = 3,
               iter = 500)
```

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
```

```
## using C compiler: 'Apple clang version 14.0.3 (clang-1403.0.22.14.1)'
## using SDK: 'MacOSX13.3.sdk'
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Framew
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/src/Core
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/src/Core
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/Core:96
## #include <complex>
##          ^~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 9e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.09 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: 0.002 seconds (Warm-up)
## Chain 1:                0.002 seconds (Sampling)
## Chain 1:                0.004 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.01 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: 0.002 seconds (Warm-up)
## Chain 2:                0.002 seconds (Sampling)
## Chain 2:                0.004 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.01 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: 0.002 seconds (Warm-up)
## Chain 3:                0.002 seconds (Sampling)
## Chain 3:                0.004 seconds (Total)
## Chain 3:
```

Print the estimates:

```
print(fit1)
```

```
## 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
```

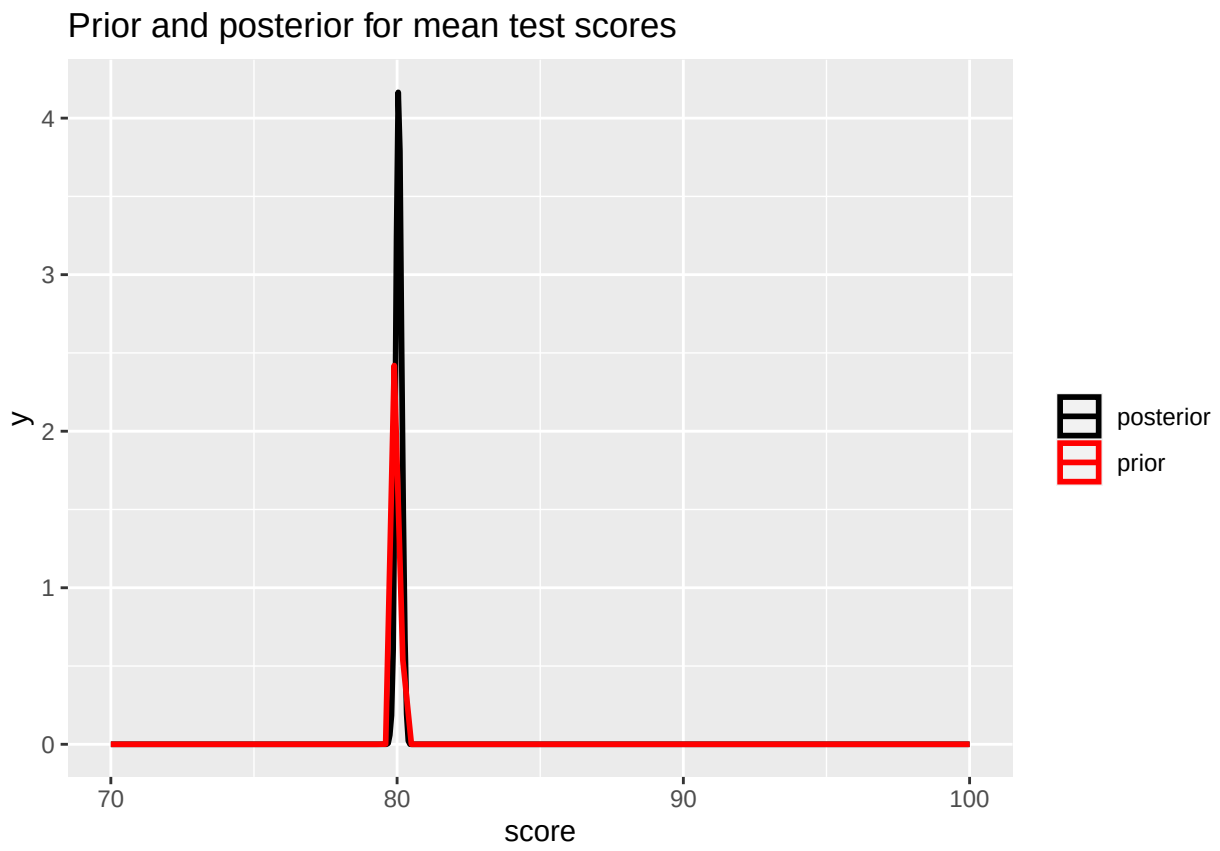
```
## mu       80.06    0.00 0.10    79.88    80.00    80.06    80.13    80.26   652
## sigma    21.44    0.03 0.67    20.26    20.95    21.42    21.88    22.87   498
## lp__  -1548.28    0.05 0.89 -1550.71 -1548.56 -1548.03 -1547.65 -1547.40   391
##       Rhat
## mu    1.00
## sigma 1.01
## lp__  1.00
##
## Samples were drawn using NUTS(diag_e) at Mon Feb 12 13:40:44 2024.
## 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 estimates changed. $\hat{\mu}$ decreased from 86 to around 80 and $\hat{\sigma}$ increased from 20 to 21.

Plot the posterior and prior distribution:

```
dsamples <- fit1 |>
  gather_draws(mu, sigma) # gather = long format

dsamples |>
  filter(.variable == "mu") |>
  ggplot(aes(.value, color = "posterior")) +
  geom_density(size = 1) +
  xlim(c(70, 100)) +
  stat_function(fun = dnorm,
                args = list(mean = mu0,
                sd = sigma0),
                aes(colour = 'prior'), size = 1) +
  scale_color_manual(name = "", values = c("prior" = "red", "posterior" = "black")) +
  ggtitle("Prior and posterior for mean test scores") +
  xlab("score")
```



The posterior mean of scores is centered around 80, which is the estimate of μ we obtained in the fitting result.

Q3

a)

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

```
X <- as.matrix(kid$mom_hs, ncol = 1) # force this to be a matrix
K <- 1
data2 <- list(y = y,
              N = length(y),
              X = X,
              K = K)

fit2 <- stan(file = "code/models/kids3.stan",
               data = data2,
               iter = 1000)
```

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: 'Apple clang version 14.0.3 (clang-1403.0.22.14.1)'
## using SDK: 'MacOSX13.3.sdk'
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Frame
## In file included from <built-in>:1:
```

```
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/src/Cor
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/src/Cor
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/Core:96
## #include <complex>
##          ^~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.43 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:                0.045 seconds (Sampling)
## Chain 1:                0.113 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.16 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: 0.085 seconds (Warm-up)
## Chain 2:                0.043 seconds (Sampling)
## Chain 2:                0.128 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 9e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.09 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: 0.058 seconds (Warm-up)
## Chain 3:                0.041 seconds (Sampling)
## Chain 3:                0.099 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.4e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.14 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: 0.068 seconds (Warm-up)
## Chain 4:                0.039 seconds (Sampling)
## Chain 4:                0.107 seconds (Total)
## Chain 4:
```

Print the estimates of the parameters:

```
print(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      77.83    0.07 1.90    74.11    76.55    77.82    79.13    81.50
## beta[1]    11.38    0.08 2.16     7.13     9.94    11.36    12.82    15.55
## sigma      19.84    0.02 0.68    18.52    19.40    19.81    20.24    21.28
## lp__    -1514.32    0.05 1.23 -1517.60 -1514.87 -1513.97 -1513.44 -1512.96
##         n_eff Rhat
## alpha     767 1.00
## beta[1]   774 1.00
## sigma    1185 1.00
## lp__      725 1.01
##
## Samples were drawn using NUTS(diag_e) at Mon Feb 12 13:41:05 2024.
## 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).
```

Now, fit the regression model using lm:

```
lm <- lm(kid_score ~ mom_hs, data=kid)
summary(lm)
```

```
##
## Call:
## lm(formula = kid_score ~ mom_hs, data = kid)
##
## 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
```

The estimate for the intercept α from lm() is around 77.5, which is close to the posterior mean of α (78.1). Similarly, the estimate for β from lm() is around 11.7, which is just slightly higher than the posterior mean of β (11.1). Thus, we confirm that the estimates of the intercept and slope are comparable to results from lm().

To estimate σ from lm(), we use the unbiased estimator as follows:

$$\hat{\sigma}^2 = \frac{SSE}{n-2}$$

```
sigma_2 <- sqrt(sum(lm$residuals^2) / (length(y) - 2))
sigma_2
```

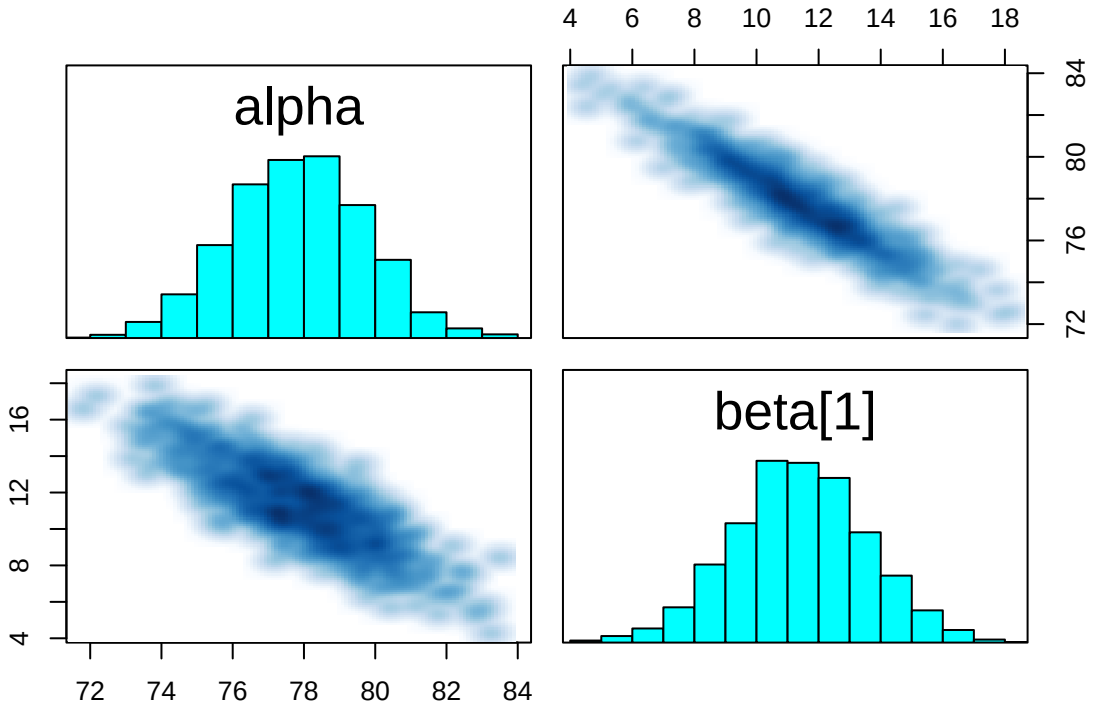
```
## [1] 19.85253
```

The estimated σ^2 from lm() is around 19.85 and the posterior mean of σ^2 is around 19.82. Thus, we confirm that they are close too.

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[1]"))
```



By observing the plot, we see a negative linear relationship between the intercept and the slope. This is a potential problem because there is collinearity in the model which will cause the model to have unstable parameter estimates.

Q4

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.

```
kid$mom_iq_centered <- kid$mom_iq - mean(kid$mom_iq)

X <- as.matrix(kid[, c("mom_hs", "mom_iq_centered")])
K <- 2

data3 <- list(y = y,
              N = length(y),
              X = X,
              K = K
)

fit3 <- stan(file = "code/models/kids3.stan",
             data = data3,
             iter = 1000)
```

```
##
```

```
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.5e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.15 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: 0.069 seconds (Warm-up)
## Chain 1:                0.051 seconds (Sampling)
## Chain 1:                0.12 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 / 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: 0.067 seconds (Warm-up)
## Chain 2:                0.048 seconds (Sampling)
## Chain 2:                0.115 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 / 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: 0.07 seconds (Warm-up)
## Chain 3:                0.051 seconds (Sampling)
## Chain 3:                0.121 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 9e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.09 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: 0.075 seconds (Warm-up)
## Chain 4:                0.049 seconds (Sampling)
## Chain 4:                0.124 seconds (Total)
## Chain 4:
```

Print the fitting result:

```
print(fit3)
```

```
## 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      82.28    0.05 1.82    78.74    81.07    82.29    83.53    85.85
## beta[1]     5.73    0.06 2.08     1.75     4.31     5.72     7.17     9.83
## beta[2]     0.56    0.00 0.06     0.45     0.52     0.56     0.61     0.69
## sigma      18.09    0.02 0.63    16.92    17.65    18.06    18.50    19.36
## lp__    -1474.44    0.05 1.43 -1477.86 -1475.17 -1474.11 -1473.38 -1472.66
##         n_eff Rhat
## alpha    1198 1.01
## beta[1]  1181 1.01
## beta[2]  1529 1.00
## sigma    1231 1.00
## lp__      786 1.00
##
## Samples were drawn using NUTS(diag_e) at Mon Feb 12 13:41:06 2024.
## 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 coefficient on the (centered) mom's IQ is estimated to be 0.56, which means on average, keeping other things constant, an unit increase in the mom's IQ will increase the kid's score by 0.56.

Q5

Now, fit the regression model using lm:

```r
lm <- lm(kid_score ~ mom_hs + mom_iq_centered, data=kid)
summary(lm)
```

```
##
## Call:
## lm(formula = kid_score ~ mom_hs + mom_iq_centered, data = kid)
##
## 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)     82.12214    1.94370  42.250  < 2e-16 ***
## mom_hs           5.95012    2.21181   2.690  0.00742 **
## mom_iq_centered  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 on the (centered) mom's IQ from lm() is also estimated to be around 0.56, thus it agrees with the bayesian estimate.

Q6

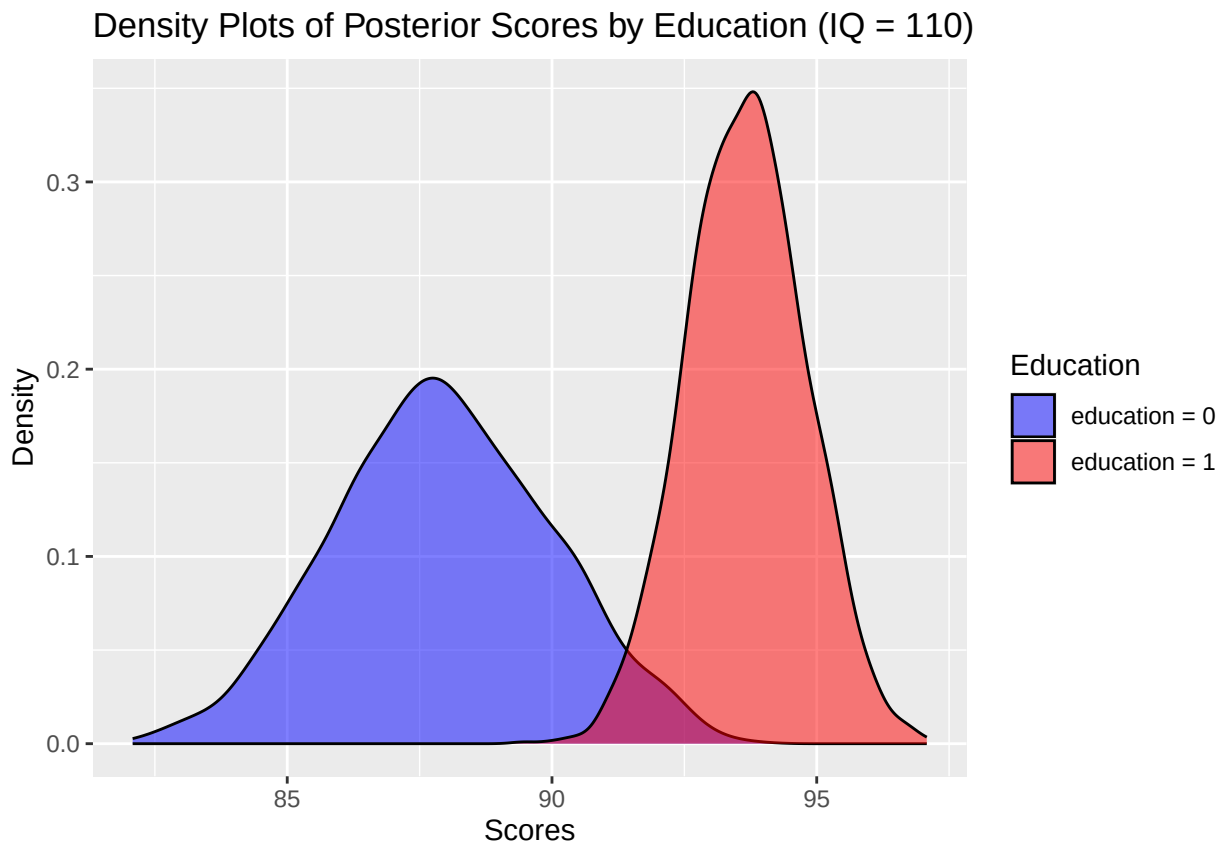
Plot the posterior estimates of scores by education of mother for mothers who have an IQ of 110.

```
# producing posterior samples
posterior_samples <- extract(fit3)

# producing posterior the estimates of scores by education
posterior_scores_hs0 <- posterior_samples[["alpha"]] + posterior_samples[["beta"]][,1] * 0 +
  posterior_samples[["beta"]][,2] * (110 - mean(kid$mom_iq))
posterior_scores_hs1 <- posterior_samples[["alpha"]] + posterior_samples[["beta"]][,1] * 1 +
  posterior_samples[["beta"]][,2] * (110 - mean(kid$mom_iq))

# plotting the estimates of scores
density_data <- data.frame(
  Scores = c(posterior_scores_hs0, posterior_scores_hs1),
  Education = rep(c("education = 0", "education = 1"), each = length(posterior_scores_hs0))
)

ggplot(density_data, aes(x = Scores, fill = Education)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plots of Posterior Scores by Education (IQ = 110)",
       x = "Scores", y = "Density") +
  scale_fill_manual(values = c("blue", "red"))
```



The density plots show that for moms who had high school education and IQ of 110, their kids' scores are

centered between 93-95, whereas for for moms who had no high school education and IQ of 110, their kids' scores are centered between 87-89.

Q7

Plot the posterior predictive distribution for a new kid with a mother who graduated high school and has an IQ of 95.

```
# producing posterior the estimates of scores
posterior_scores <- posterior_samples[["alpha"]] + posterior_samples[["beta"]][,1] * 1 +
  posterior_samples[["beta"]][,2] * (95 - mean(kid$mom_iq)) + posterior_samples[["sigma"]]

# plotting the scores
hist(posterior_scores, main = "Posterior Predictive Distribution for a New Kid",
     xlab = "Predicted Scores")
```