STA2201_Lab5

Alice Huang

08/02/2023

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).

The data look like this:

```
kidiq <- read_rds(here("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.

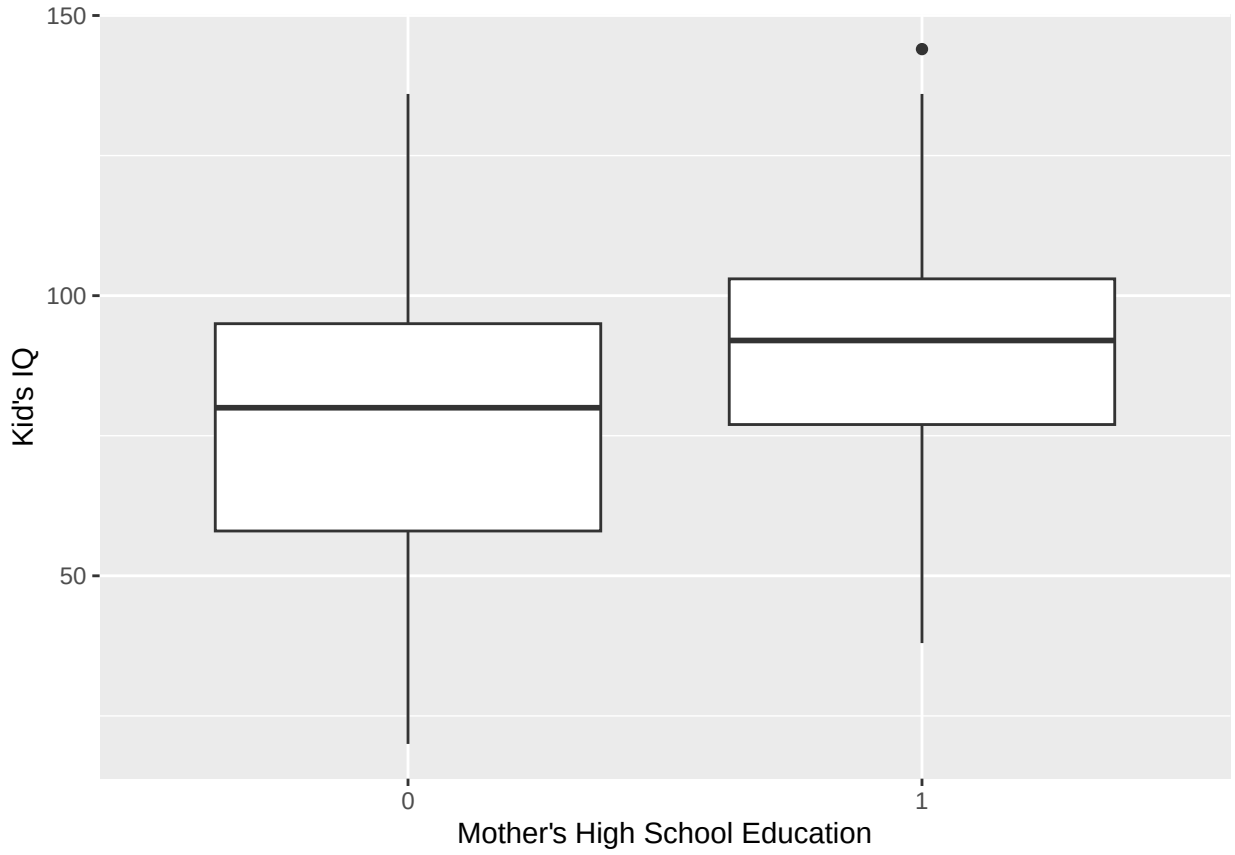
Descriptives

Question 1

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

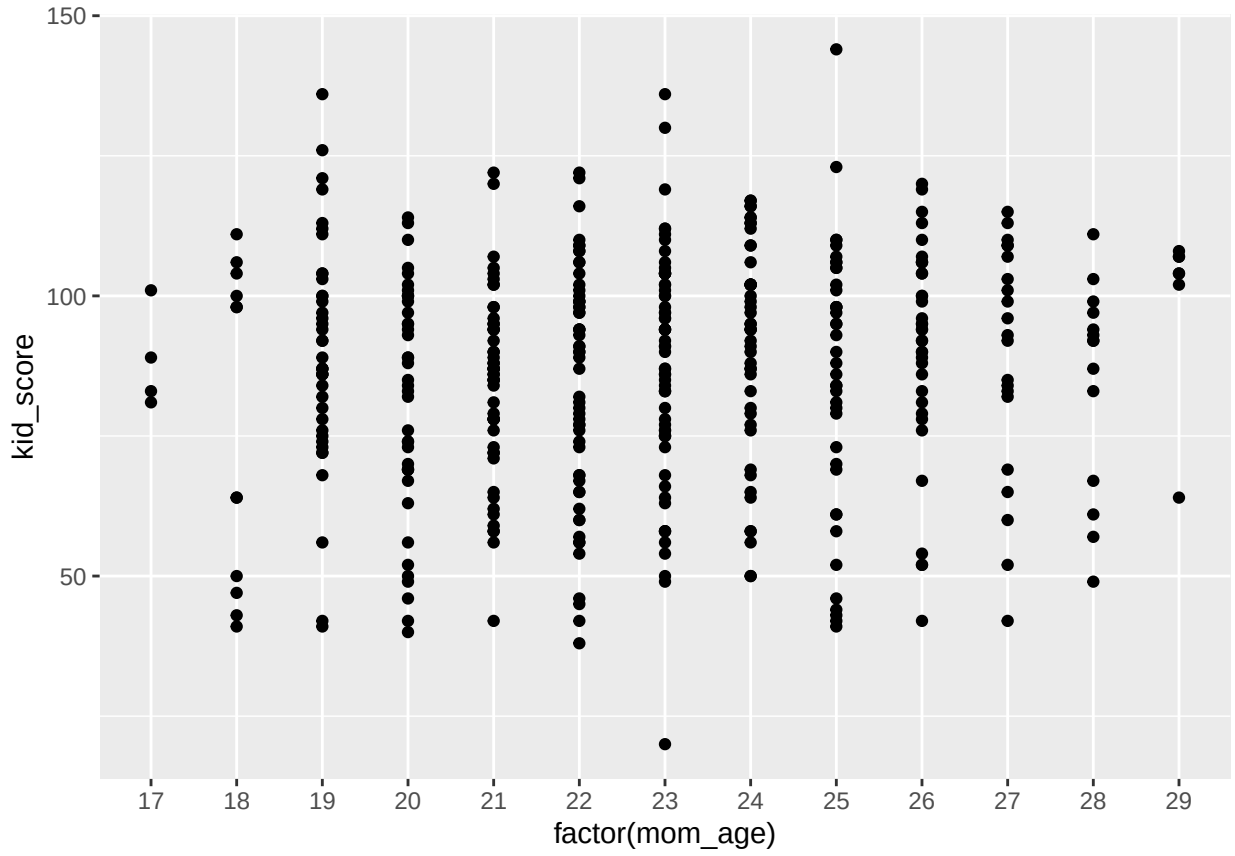
- Explain what your graph/ tables show
- Choose a graph type that's appropriate to the data type

```
kidiq %>% ggplot(aes(x=factor(mom_hs), y = kid_score)) + geom_boxplot() + labs(x="Mother's High School
```



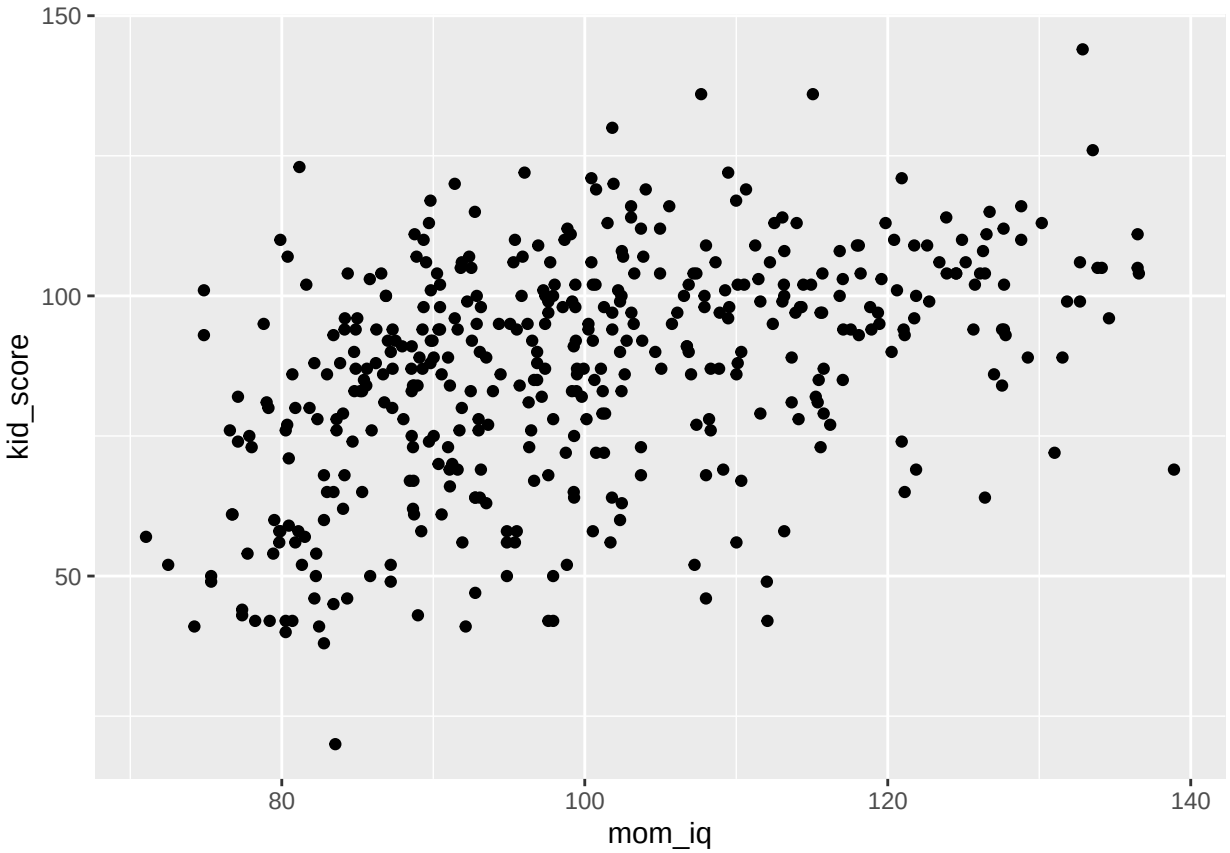
We can see that the median IQ score for children of mothers with high school education was slightly higher than the median IQ score for children of mothers without high school education.

```
kidiq %>% ggplot(aes(x=factor(mom_age), y=kid_score)) + geom_point()
```



There does not appear to be a significant change in the kids' IQ scores as the mother's age increases.

```
kidiq %>% ggplot(aes(x=mom_iq, y=kid_score)) + geom_point()
```



It seems that there is positive correlation between the mother's IQ score and the kid's IQ score.

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 = here("kids2.stan"),
            data = data,
            chains = 3,
            iter = 500)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 5.2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.52 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: 0.018 seconds (Warm-up)
## Chain 1:                0.01 seconds (Sampling)
## Chain 1:                0.028 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.2e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.12 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: 0.036 seconds (Warm-up)
## Chain 2:                0.011 seconds (Sampling)
## Chain 2:                0.047 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 / 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: 0.013 seconds (Warm-up)
## Chain 3:                0.013 seconds (Sampling)
## Chain 3:                0.026 seconds (Total)
## Chain 3:
```

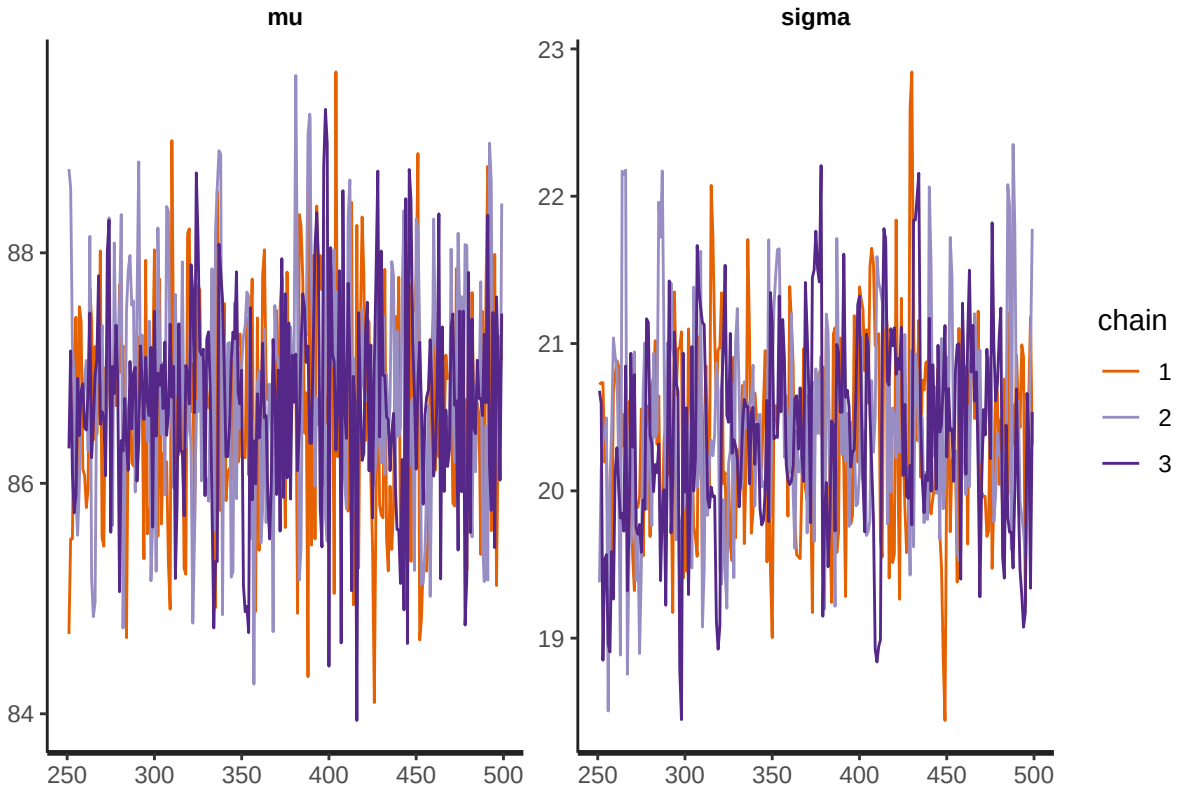
Look at the summary

```
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       86.72    0.04 0.95    84.87    86.08    86.72    87.33    88.65   675
## sigma    20.39    0.04 0.68    19.08    19.93    20.38    20.85    21.79   359
## lp__  -1525.73    0.05 0.95 -1528.29 -1526.12 -1525.44 -1525.04 -1524.78   305
##       Rhat
## mu    1.00
## sigma 1.00
## lp__  1.01
##
## Samples were drawn using NUTS(diag_e) at Mon Feb 13 22:18:45 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).
```

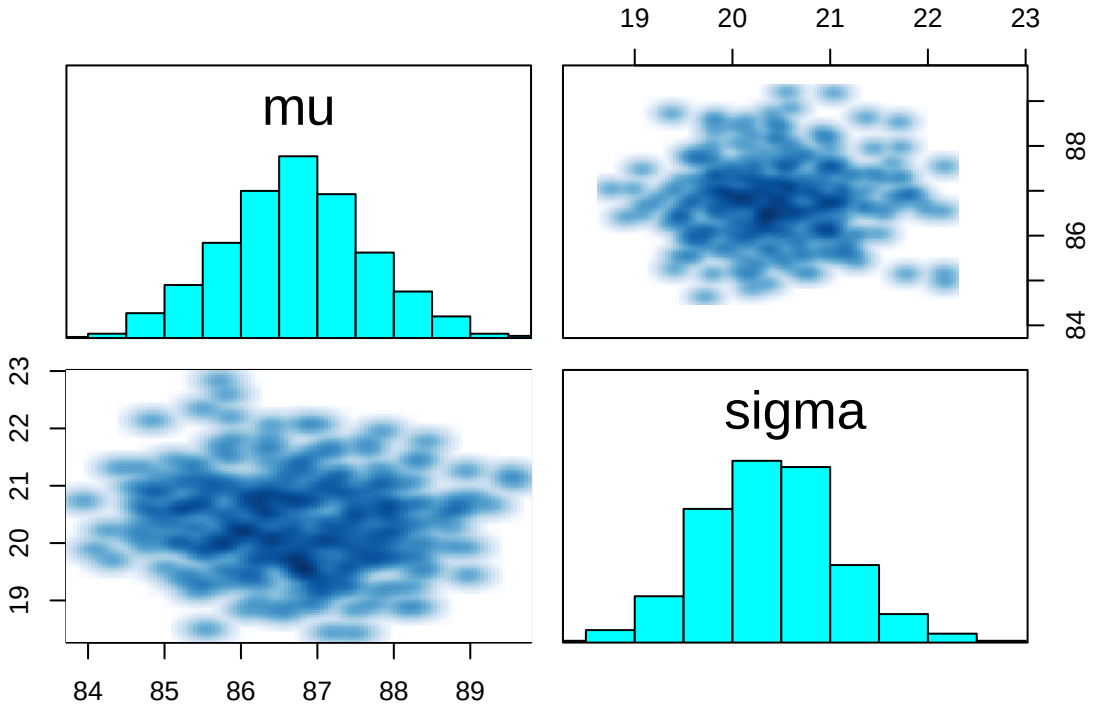
Traceplot

```
traceplot(fit)
```

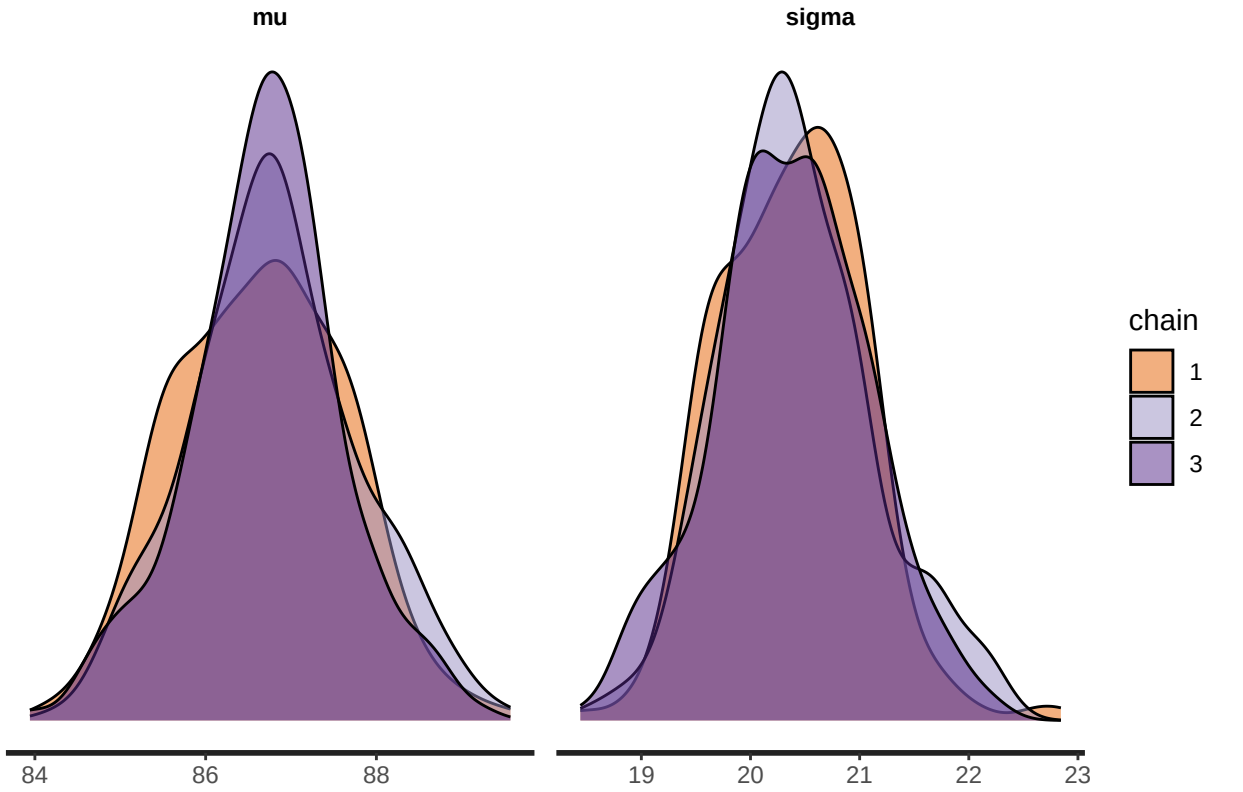


All looks fine.

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




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



Understanding output

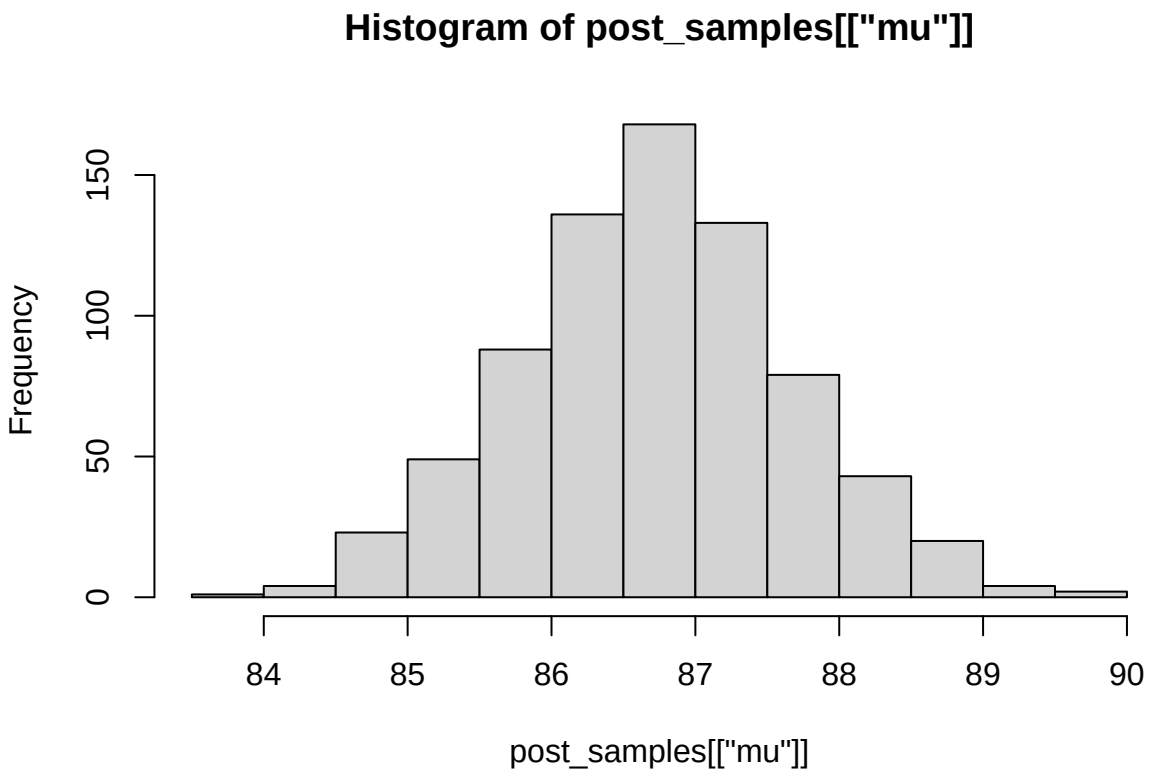
What does the model actually give us? A number of samples from the posteriors. To see this, we can use `extract` to get the samples.

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

```
## [1] 84.92180 85.96946 87.80234 87.43530 85.21070 85.97436
```

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] 86.72395
```

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

```
##    2.5%
## 84.8686
```

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

```
##    97.5%
## 88.64942
```

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:

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

```
## # A tibble: 1,500 x 5
## # Groups:   .variable [2]
##    .chain .iteration .draw .variable .value
##     <int>      <int> <int> <chr>      <dbl>
##  1      1          1     1 mu          84.7
##  2      1          2     2 mu          85.5
##  3      1          3     3 mu          85.5
##  4      1          4     4 mu          86.9
##  5      1          5     5 mu          87.4
##  6      1          6     6 mu          85.8
##  7      1          7     7 mu          87.5
##  8      1          8     8 mu          87.4
##  9      1          9     9 mu          86.1
## 10      1         10    10 mu          86.1
## # ... with 1,490 more rows
```

```
# 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  84.7  20.7
##  2      1          2     2  85.5  20.7
##  3      1          3     3  85.5  20.7
##  4      1          4     4  86.9  20.2
##  5      1          5     5  87.4  20.3
##  6      1          6     6  85.8  19.6
##  7      1          7     7  87.5  19.9
##  8      1          8     8  87.4  19.5
##  9      1          9     9  86.1  20.7
## 10      1         10    10  86.1  20.8
## # ... with 740 more rows
```

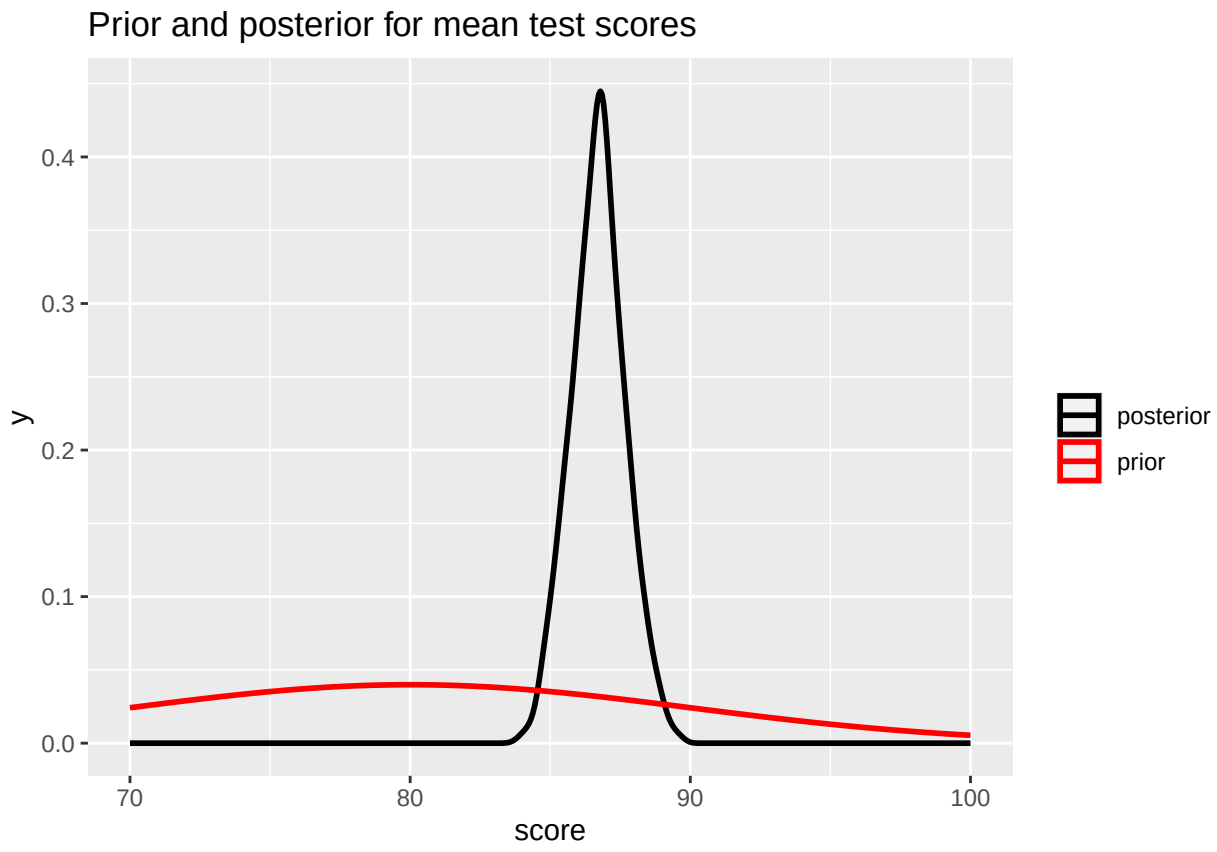
```
# quickly calculate the quantiles using

dsamples %>%
  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   85.5   87.9    0.8 median qi
## 2 sigma       20.4   19.6   21.2    0.8 median qi
```

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

```r
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")
```



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.

```r
y <- kidiq$kid_score
mu0 <- 80
sigma01 <- 0.1

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

```
             mu0 = mu0,
             sigma0 = sigma01)

fit1 <- stan(file = here("C:/Users/Alice/Desktop/GRAD SCHOOL/STA2201/STA2201AppliedStatAliceHuang/kids2
            data = data,
            chains = 3,
            iter = 500)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.1 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: 0.015 seconds (Warm-up)
## Chain 1:                0.015 seconds (Sampling)
## Chain 1:                0.03 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.11 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: 0.018 seconds (Warm-up)
## Chain 2:                0.027 seconds (Sampling)
## Chain 2:                0.045 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 / 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: 0.021 seconds (Warm-up)
## Chain 3:                0.013 seconds (Sampling)
## Chain 3:                0.034 seconds (Total)
## Chain 3:
```

```
dsamples2 <- fit1  %>%
  gather_draws(mu, sigma) # gather = long format

# quickly calculate the quantiles using

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

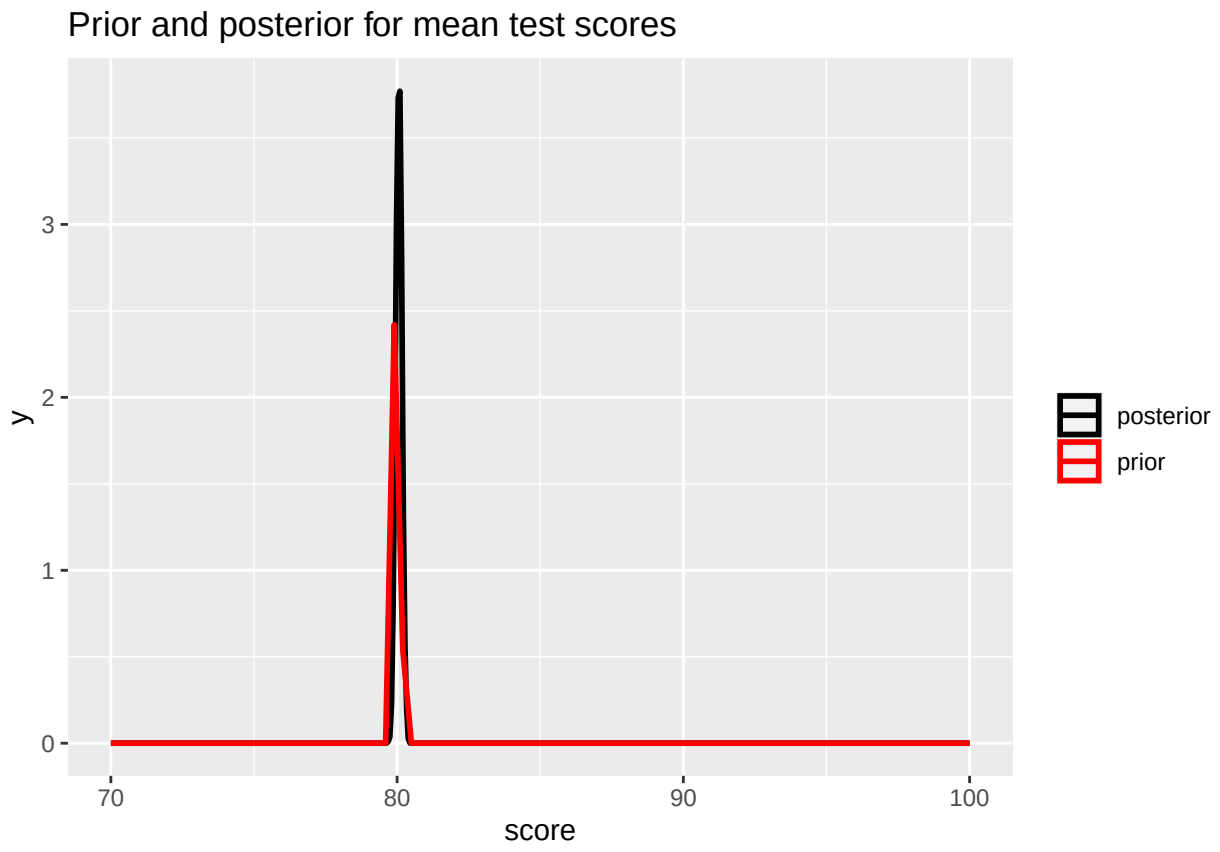
```
## # A tibble: 2 x 7
##   .variable .value .lower .upper .width .point .interval
##   <chr>      <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
## 1 mu          86.7   85.5   87.9    0.8 median qi
## 2 sigma       20.4   19.6   21.2    0.8 median qi
```

```
dsamples2 %>%
  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          80.1   79.9   80.2    0.8 median qi
## 2 sigma       21.4   20.5   22.4    0.8 median qi
```

The estimate for mu changed from 86.69947 to 80.07737. The estimate for sigma changed from 20.31330 to 21.37362.

```
dsamples2 %>%
  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 = sigma01),
        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")
```



As we can see from the plot, the plot of the posterior and the plot of the prior are closer to each other than they previously were when the standard deviation was 10.

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("kids3.stan"),
             data = data,
             iter = 1000)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000525 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 5.25 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: 0.211 seconds (Warm-up)
## Chain 1:                0.104 seconds (Sampling)
## Chain 1:                0.315 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2.3e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.23 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: 0.255 seconds (Warm-up)
## Chain 2:                0.099 seconds (Sampling)
## Chain 2:                0.354 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2.3e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.23 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: 0.148 seconds (Warm-up)
## Chain 3:                0.098 seconds (Sampling)
## Chain 3:                0.246 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 3.6e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.36 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: 0.242 seconds (Warm-up)
## Chain 4:                0.11 seconds (Sampling)
## Chain 4:                0.352 seconds (Total)
## Chain 4:
```

Question 3

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

```
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      77.99    0.08 1.99    73.93    76.66    78.07    79.31    81.86
## beta[1]    11.25    0.08 2.20     6.96     9.74    11.17    12.71    15.65
## sigma      19.83    0.02 0.68    18.52    19.38    19.83    20.28    21.19
## lp__    -1514.34    0.05 1.24 -1517.53 -1514.87 -1514.00 -1513.46 -1512.96
##         n_eff Rhat
## alpha     651 1.00
## beta[1]   678 1.00
## sigma    1103 1.00
## lp__      719 1.01
##
## Samples were drawn using NUTS(diag_e) at Mon Feb 13 22:21:18 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).
```

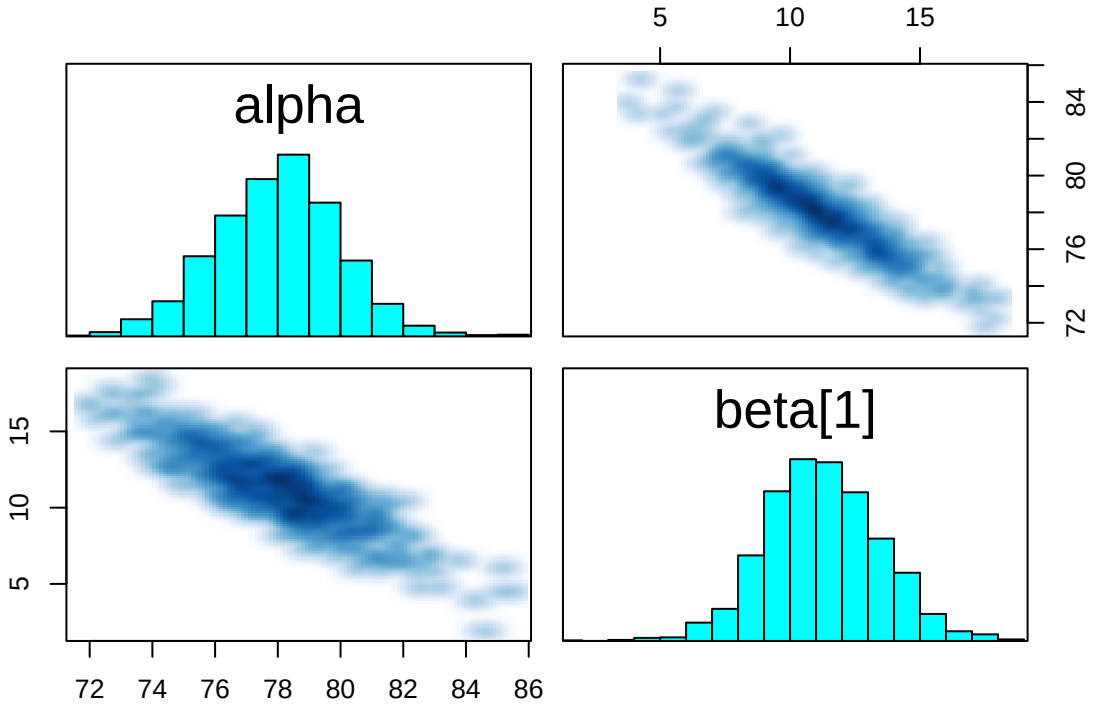
```
summary(lm(kid_score ~ mom_hs, data = kidiq))
```

```
##
## 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
```

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"))
```



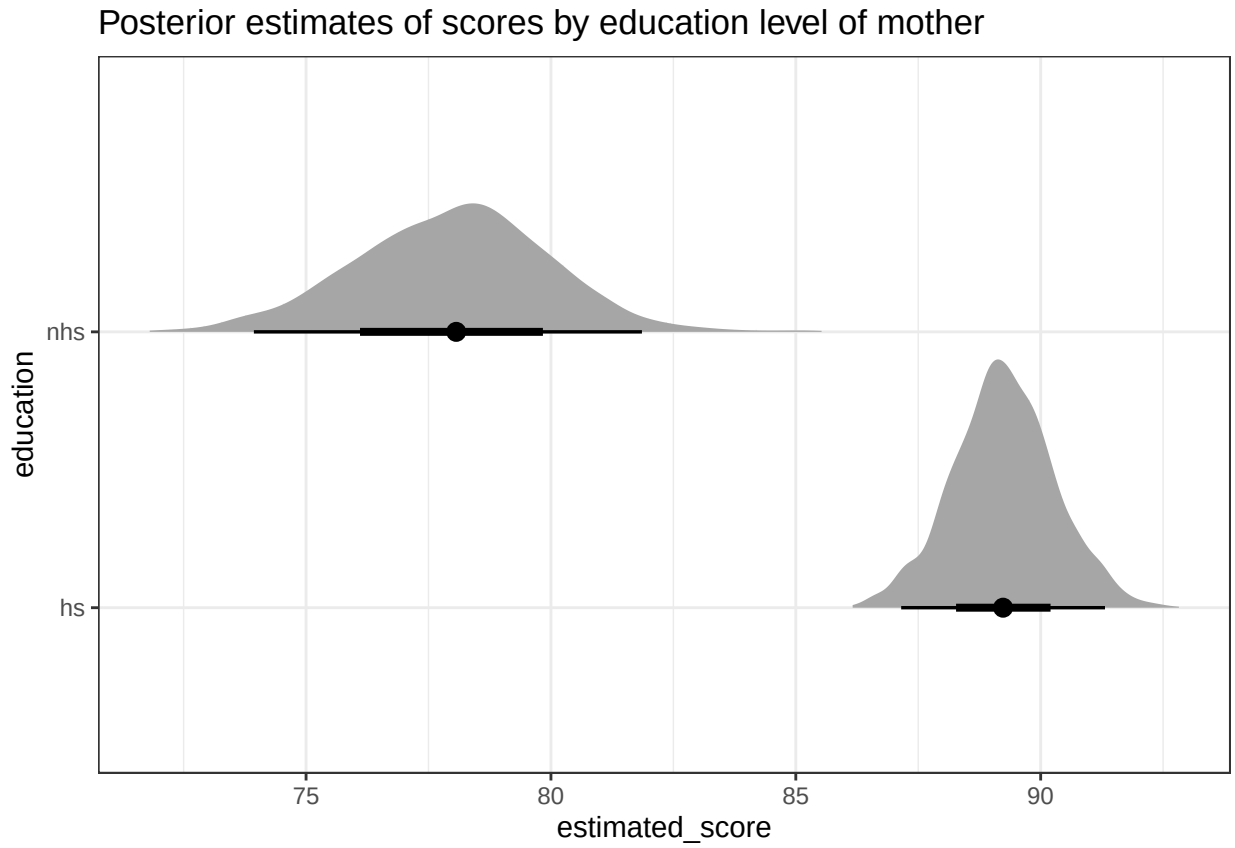
There seems to be strong correlation between alpha and beta, the slope and the intercept.

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 %>% mutate(mom_iq_cent = mom_iq - mean(mom_iq)) -> kidiq

X <- cbind(as.matrix(kidiq$mom_hs), as.matrix(kidiq$mom_iq_cent)) # force this to be a matrix
K <- 2

data <- list(y = y, N = length(y),
             X =X, K = K)
fit4 <- stan(file = here("kids3.stan"),
             data = data,
             iter = 1000)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.44 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: 0.403 seconds (Warm-up)
## Chain 1:                0.185 seconds (Sampling)
## Chain 1:                0.588 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 3.4e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.34 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: 0.334 seconds (Warm-up)
## Chain 2:                0.208 seconds (Sampling)
## Chain 2:                0.542 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3.2e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.32 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: 0.313 seconds (Warm-up)
## Chain 3:                0.14 seconds (Sampling)
## Chain 3:                0.453 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.8e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.28 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: 0.214 seconds (Warm-up)
## Chain 4:                0.127 seconds (Sampling)
## Chain 4:                0.341 seconds (Total)
## Chain 4:
```

```
fit4
```

```
## 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      82.36    0.06 1.92    78.57    81.08    82.36    83.69    86.01
## beta[1]     5.66    0.07 2.17     1.45     4.15     5.59     7.17    10.08
## beta[2]     0.57    0.00 0.06     0.45     0.53     0.57     0.61     0.68
## sigma      18.11    0.02 0.60    17.00    17.70    18.09    18.48    19.35
## lp__    -1474.37    0.05 1.37 -1477.79 -1475.06 -1474.06 -1473.37 -1472.64
```

```
##         n_eff Rhat
## alpha    1000    1
## beta[1]   990    1
## beta[2]  1333    1
## sigma    1522    1
## lp__      808    1
##
## Samples were drawn using NUTS(diag_e) at Mon Feb 13 22:21:23 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).
```

When the mom's mean-centred IQ increases by 1 unit, the kid's IQ is expected to increase by 0.57 units.

Question 5

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

```
summary(lm(kid_score ~ mom_hs + mom_iq_cent, data = kidiq))
```

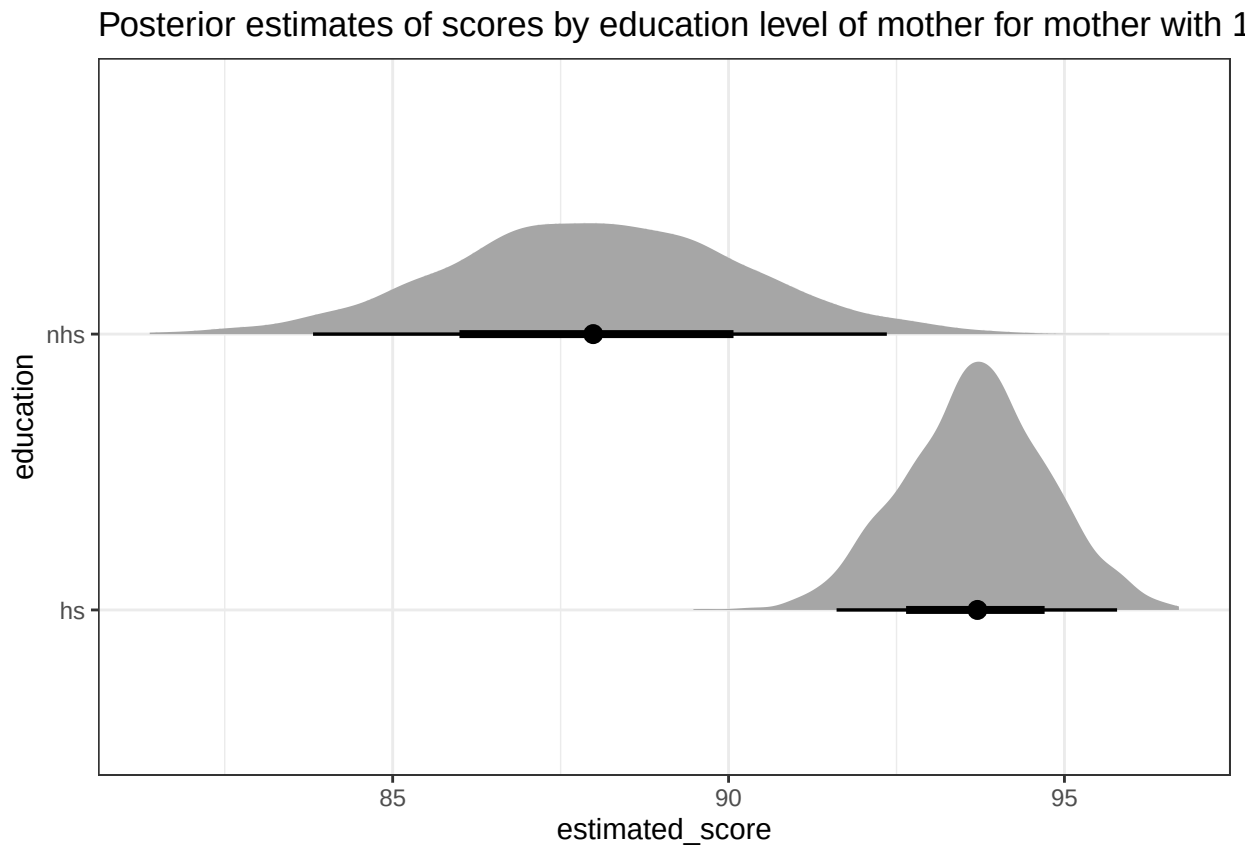
```
##
## Call:
## lm(formula = kid_score ~ mom_hs + mom_iq_cent, data = kidiq)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.873 -12.663   2.404  11.356  49.545
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 82.12214    1.94370  42.250  < 2e-16 ***
## mom_hs       5.95012    2.21181   2.690  0.00742 **
## mom_iq_cent  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
```

Question 6

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

```
fit4 %>%
  spread_draws(alpha, beta[k], sigma) %>%
  pivot_wider(names_from = k, values_from = beta) %>%
  rename(beta_1 = `1`, beta_2 = `2`) %>%
  # add effect of mom's iq being 110
     mutate(nhs = alpha + beta_2*(110 - mean(kidiq$mom_iq)),
          hs = alpha + beta_1 + beta_2*(110 - mean(kidiq$mom_iq))) %>%
```

```
  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 for mother with 110 IQ")
```



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.

```
post_samples <- extract(fit4)
new_mom_hs <- 1
new_mom_iq <- 95 - mean(kidiq$mom_iq)
#new_mom <- as.matrix(c(new_mom_hs, new_mom_iq), ncol = 2)

alpha_hat <- post_samples[["alpha"]]
beta_hat_1 <- post_samples[["beta"]][,1]
beta_hat_2 <- post_samples[["beta"]][,2]

preds <- alpha_hat + beta_hat_1*new_mom_hs + beta_hat_2*new_mom_iq

sigma_hat <- post_samples[["sigma"]]
```

```
ppd <- rnorm(n=length(sigma_hat), mean = preds, sd = sigma_hat)

data.frame(ppd) %>% ggplot(aes(x=ppd)) + geom_histogram() + labs(x="Posterior Predictive distribution")
```