Week 5: Bayesian linear regression and introduction to Stan

12/02/23

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 <- read_rds(here("data","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.

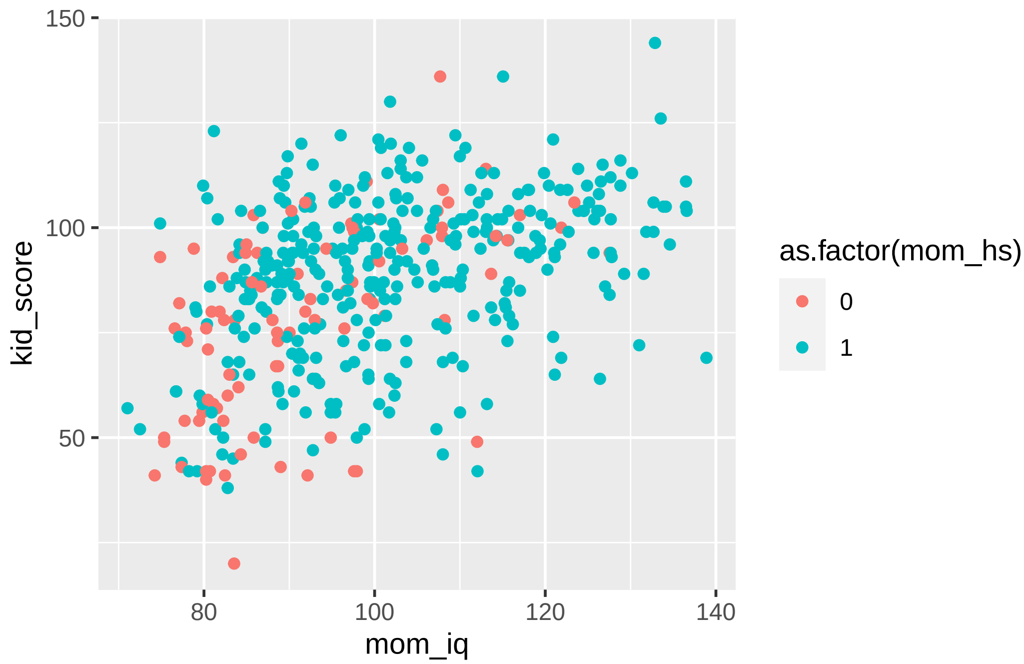
Descriptives

Question 1

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

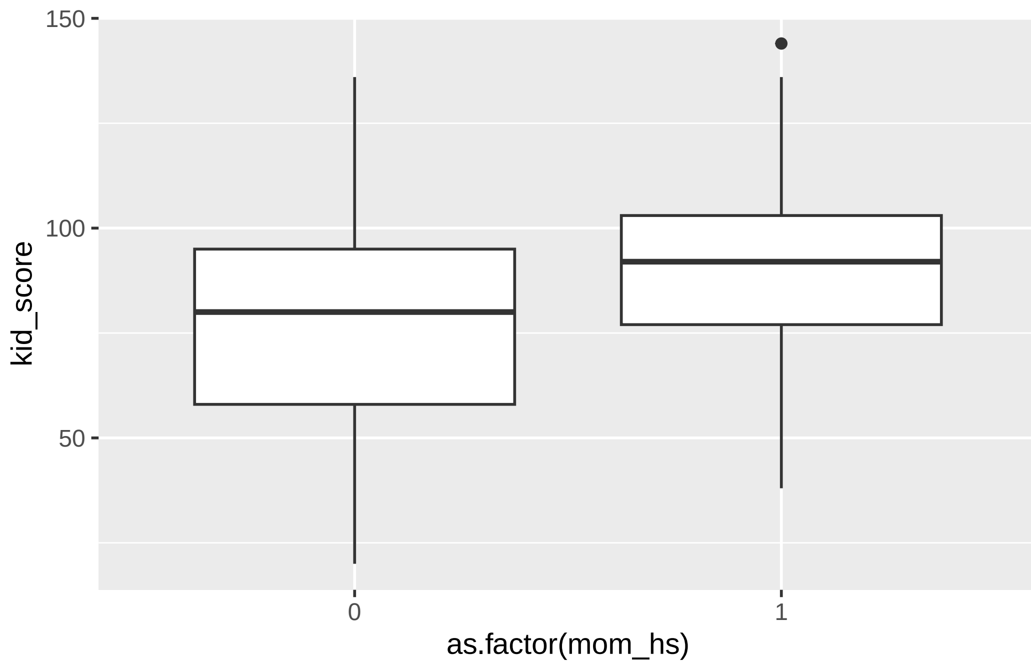
- Explain what your graph/ tables show
- Choose a graph type that's appropriate to the data type

```
kidiq %>%
ggplot(aes(x = mom_iq, y = kid_score, col = as.factor(mom_hs))) +
geom_point()
```



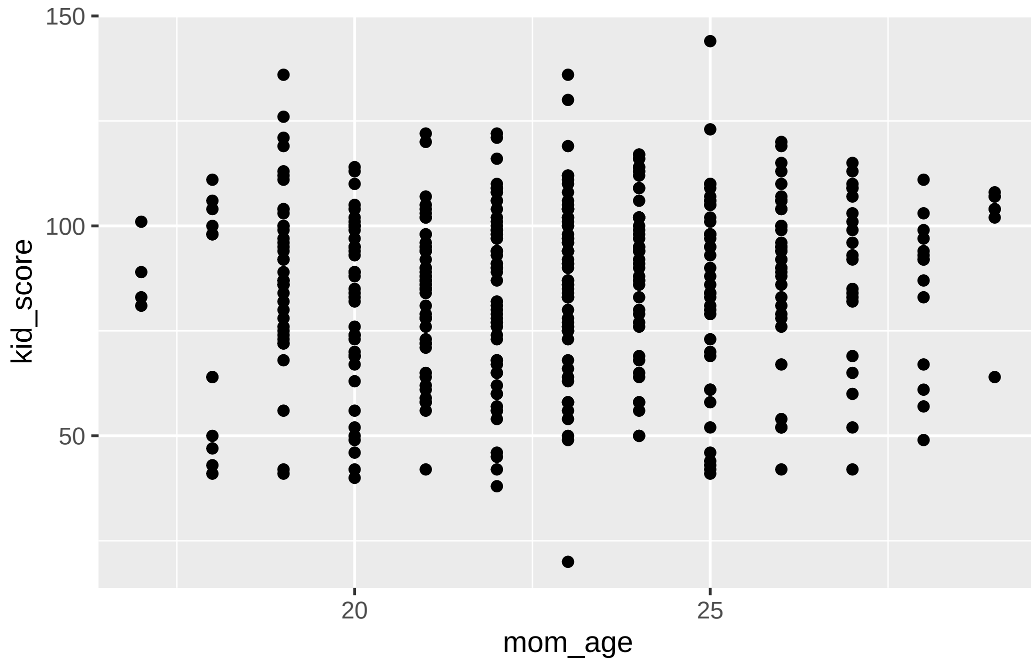
The plot shows that there is much more observations with mom attended high school. It looks like there is increasing trend for kid's score as mom's iq increase no matter the mom attended high school or not.

```
kidiq %>%
ggplot(aes(x = as.factor(mom_hs), y = kid_score)) +
geom_boxplot()
```



The box plot shows that there is wider variability in kid's score when mother did not attend high school. And when the mom didn't attend high school, the median, 25% and 75% quantile of kid's score are lower.

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



Based on the scatter plot, I don't think mom's age affect kid's 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:

```r
fit <- stan(file = here("Labs/Lab5/kids2.stan"),
            data = data,
            chains = 3,
            iter = 500)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1: 
Chain 1: Gradient evaluation took 2.7e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.27 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: 0.008 seconds (Warm-up)
Chain 1:                0.003 seconds (Sampling)
Chain 1:                0.011 seconds (Total)
Chain 1: 

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2: 
Chain 2: Gradient evaluation took 4e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.04 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: 0.009 seconds (Warm-up)
Chain 2:                0.005 seconds (Sampling)
Chain 2:                0.014 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 4e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.04 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: 0.008 seconds (Warm-up)
Chain 3:                0.004 seconds (Sampling)
Chain 3:                0.012 seconds (Total)
Chain 3:
```

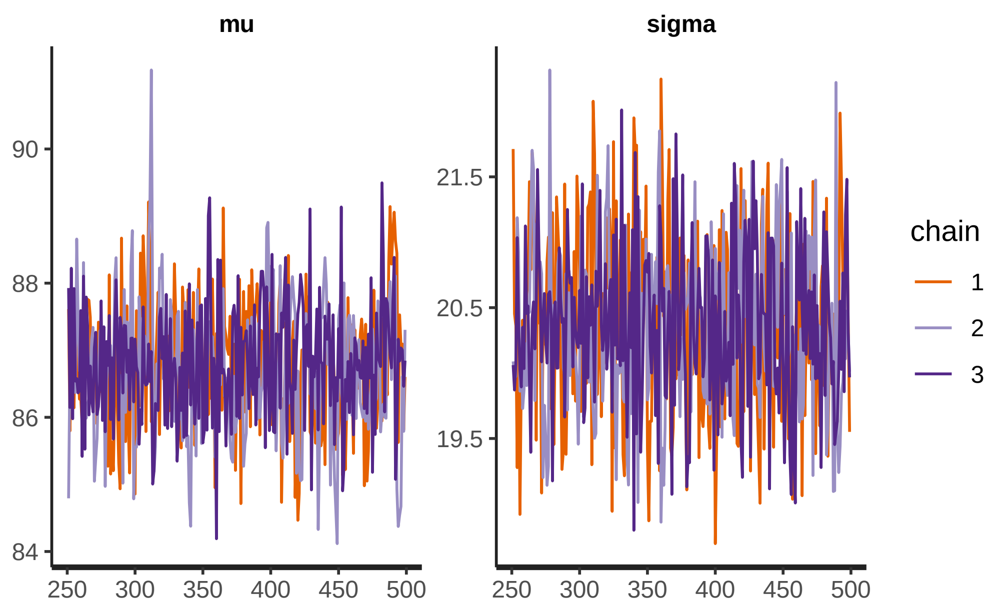
Look at the summary

```
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      86.76    0.04 0.95   84.93   86.10   86.76   87.42   88.63   499
sigma   20.38    0.02 0.66   19.14   19.93   20.39   20.84   21.65   694
lp__ -1525.69    0.05 0.95 -1528.01 -1526.04 -1525.43 -1525.04 -1524.79   409
      Rhat
mu    1.00
sigma 1.00
lp__  1.01

Samples were drawn using NUTS(diag_e) at Sun Feb 12 22:46:38 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).
```

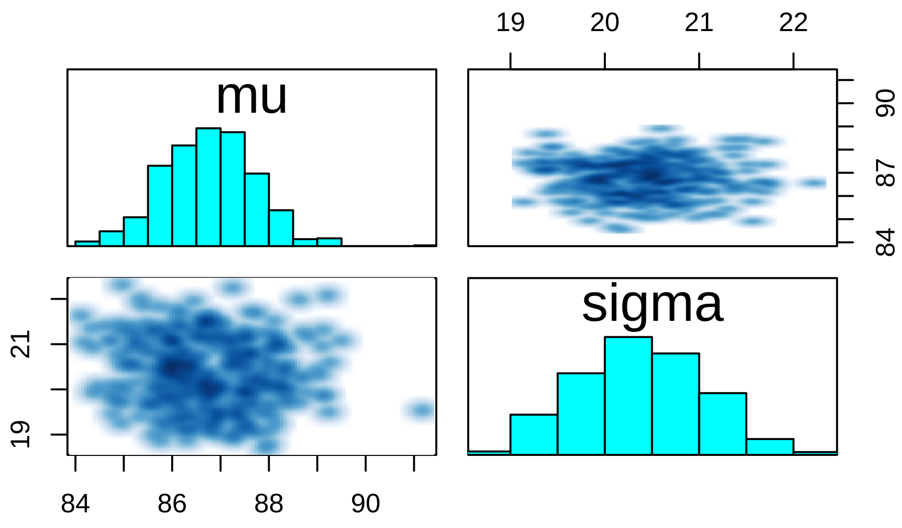
Traceplot

```
traceplot(fit)
```

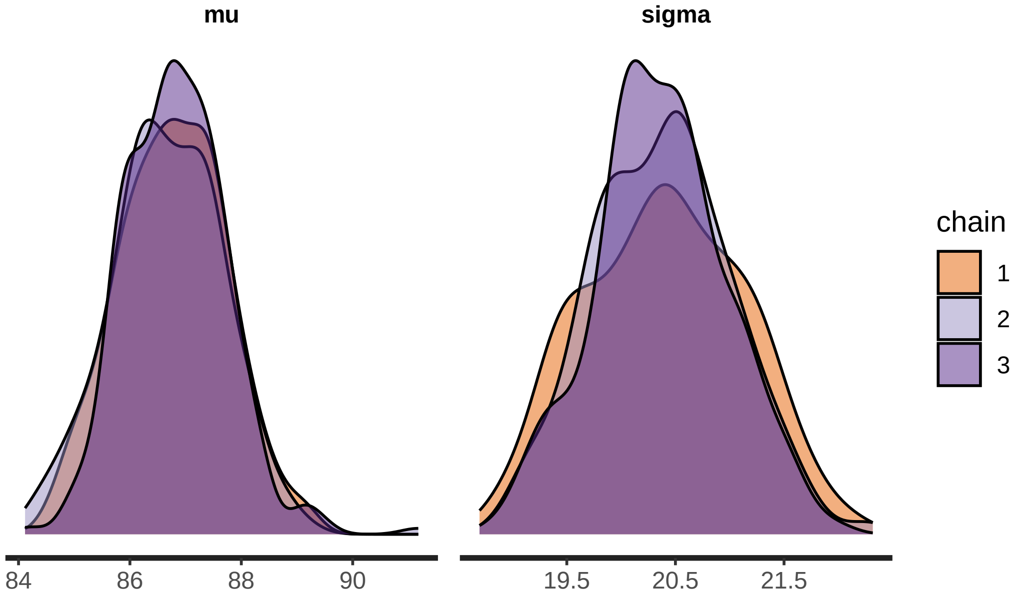


All looks fine.

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




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



Understanding output

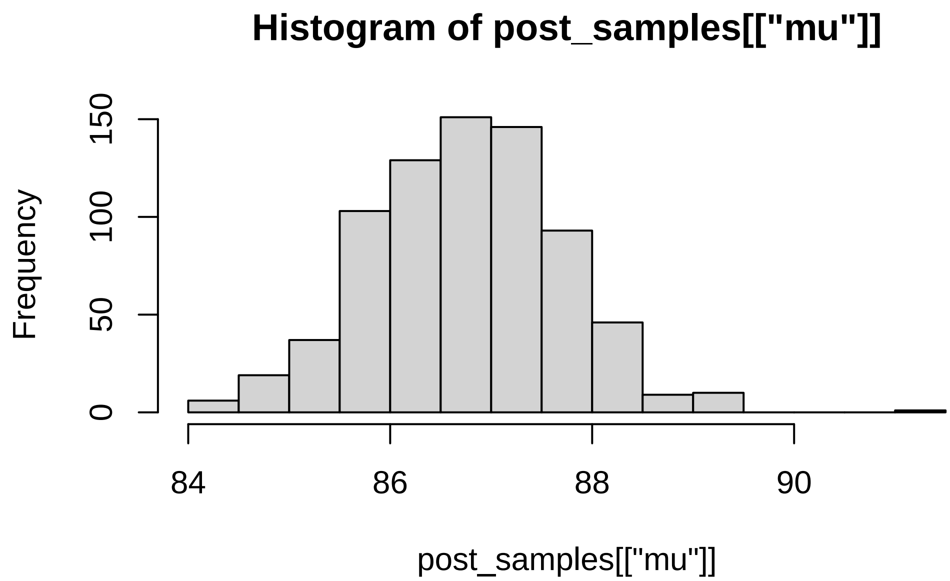
What does the model actually give us? A number of samples from the posteriors. To see this, we can use `extract` to get the samples.

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

```
[1] 87.29974 87.25138 85.90792 87.89547 86.71577 87.38412
```

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"]])
```




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

```
[1] 86.75602
```

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

```
    2.5% 
84.92725 
```

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

```
   97.5% 
88.63258 
```

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          87.6
 2      1          2     2 mu          85.8
 3      1          3     3 mu          86.8
 4      1          4     4 mu          86.9
 5      1          5     5 mu          86.1
 6      1          6     6 mu          87.3
 7      1          7     7 mu          86.8
 8      1          8     8 mu          86.7
 9      1          9     9 mu          86.3
10      1         10    10 mu          86.3
# ... 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  87.6  21.7
 2      1          2     2  85.8  20.5
 3      1          3     3  86.8  20.4
 4      1          4     4  86.9  19.3
 5      1          5     5  86.1  20.6
```

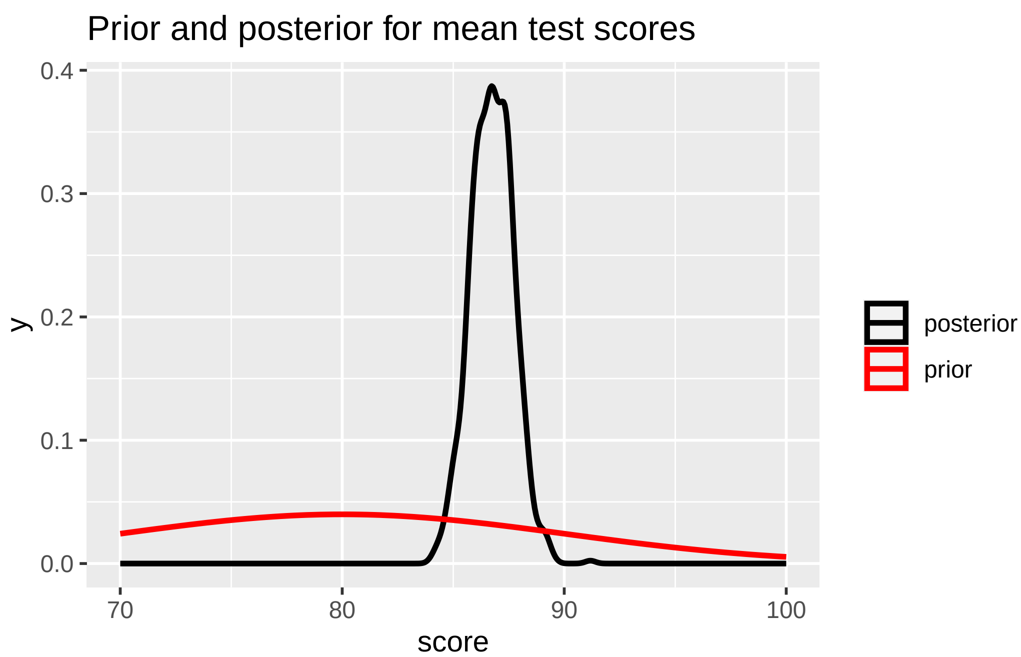
```
 6       1           6     6  87.3  18.9
 7       1           7     7  86.8  19.9
 8       1           8     8  86.7  20.4
 9       1           9     9  86.3  20.2
10       1          10    10  86.3  20.2
# ... with 740 more rows
```

```
# 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          86.8   85.6   87.9    0.8 median qi
2 sigma       20.4   19.5   21.2    0.8 median qi
```

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

```
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.

```
y <- kidiq$kid_score
mu0 <- 80
sigma0 <- 0.1

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

Now we can run the model:

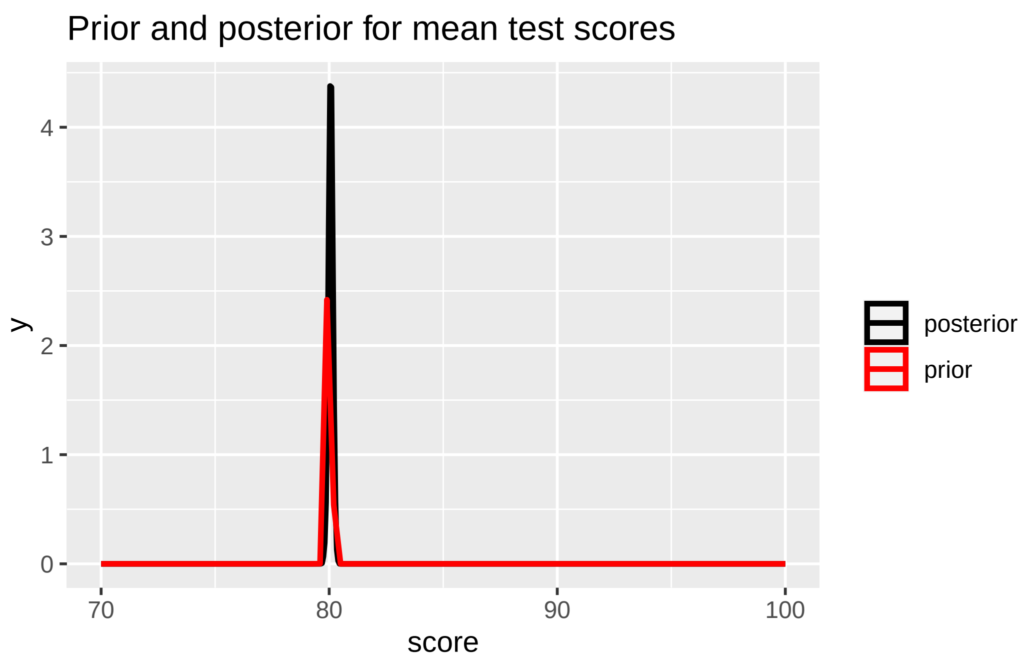
```
fit <- stan(file = here("Labs/Lab5/kids2.stan"),
            data = data,
            chains = 3,
            iter = 500)
```

```
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 / 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: 0.004 seconds (Warm-up)
Chain 1:                0.004 seconds (Sampling)
Chain 1:                0.008 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 5e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.05 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:                0.003 seconds (Sampling)
Chain 2:                0.007 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 6e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.06 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:                0.003 seconds (Sampling)
Chain 3:                0.008 seconds (Total)
Chain 3:
```

```r
dsamples <- fit  |>
  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")
```




```
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       80.06    0.00 0.09    79.88    80.00    80.07    80.13    80.25   570
sigma    21.37    0.03 0.71    20.05    20.89    21.36    21.83    22.77   617
lp__  -1548.29    0.05 0.92 -1550.86 -1548.66 -1547.99 -1547.65 -1547.38   329
      Rhat
mu    1.01
sigma 1.00
lp__  1.00

Samples were drawn using NUTS(diag_e) at Sun Feb 12 22:46:41 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 estimates does change. The mean of the posterior of mu became 80.06, very close to our strong prior that the mean should be around 80. The sd of posterior of mu became 0.1, much smaller than before(0.93). This also reflected on the posterior density plot that we have much narrower density. Also, the sigma increased slightly from 20.40 to 21.41.

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("Labs/Lab5/kids3.stan"),
            data = data,
            iter = 1000)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 4.2e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.42 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: 0.13 seconds (Warm-up)
Chain 1:                0.075 seconds (Sampling)
Chain 1:                0.205 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 2.2e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.22 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: 0.114 seconds (Warm-up)
Chain 2:                0.07 seconds (Sampling)
Chain 2:                0.184 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
```

```
Chain 3:
Chain 3: Gradient evaluation took 1.9e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.19 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: 0.13 seconds (Warm-up)
Chain 3:                0.061 seconds (Sampling)
Chain 3:                0.191 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 1.5e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.15 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: 0.105 seconds (Warm-up)
Chain 4:                0.066 seconds (Sampling)
Chain 4:                0.171 seconds (Total)
Chain 4:
```

Question 3

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

```r
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      78.00    0.08 2.03    74.19    76.69    77.94    79.29    82.03
beta[1]    11.19    0.09 2.28     6.62     9.75    11.29    12.66    15.57
sigma      19.84    0.02 0.67    18.59    19.36    19.80    20.28    21.19
lp__    -1514.37    0.05 1.28 -1517.77 -1514.93 -1514.04 -1513.45 -1512.97
        n_eff Rhat
alpha     614 1.01
beta[1]   659 1.01
sigma    1148 1.00
lp__      733 1.00

Samples were drawn using NUTS(diag_e) at Sun Feb 12 22:47:38 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).
```

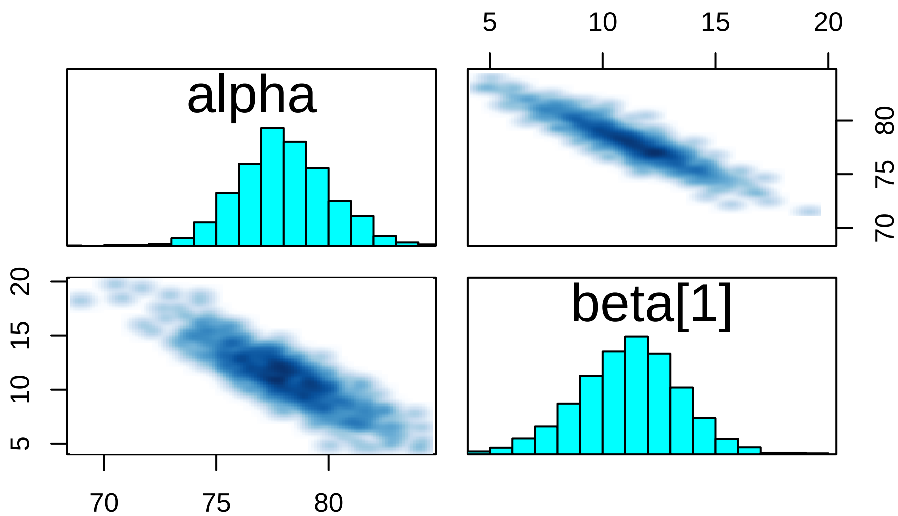
```r
model3 = lm(kid_score ~ mom_hs, data = kidiq)
summary(model3)
```

```
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
```

The output from lm agrees with our model output. In lm, we have intercept of 77.548 and estimate for mom_hs is 11.771, where from our stan model, we have intercept of 77.96 and estimate for mom_hs is 11.21.

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

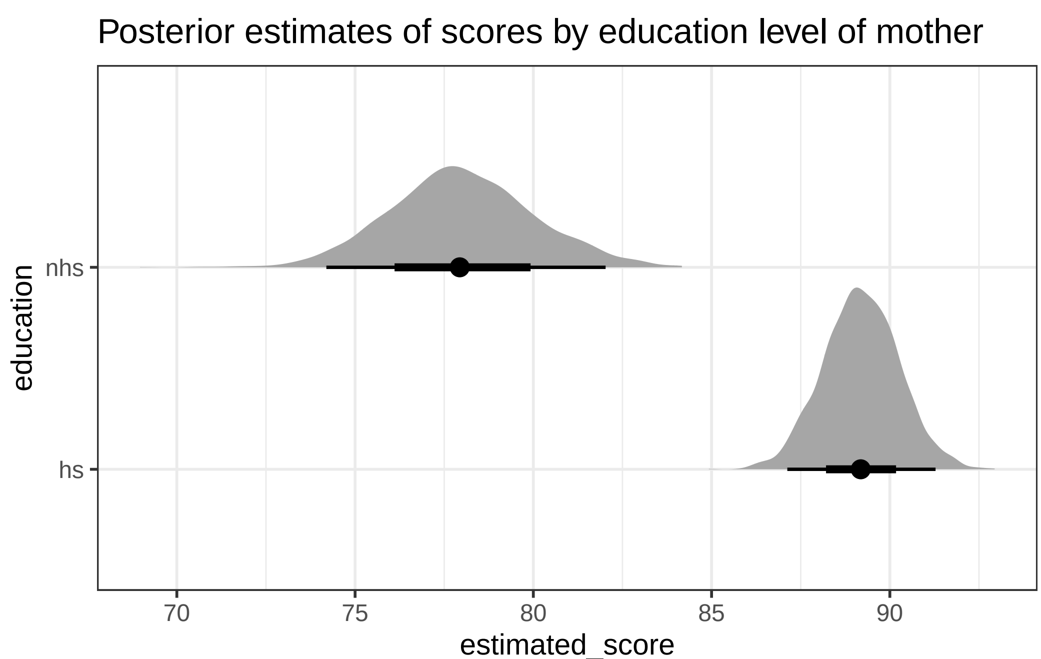


Yes, there is a correlction problem that as alpha(intercept) increase we get a lower beta(slope).

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.

```
kidiqdata <- kidiq %>% mutate(ceneterdiq = scale(mom_iq, scale = FALSE))
X <- as.matrix(cbind(kidiqdata$mom_hs, kidiqdata$ceneterdiq), nrow = 434, ncol = 2) # forc
K <- 2

data <- list(y = y, N = length(y),
             X =X, K = K)
fit3 <- stan(file = here("Labs/Lab5/kids3.stan"),
            data = data,
            iter = 1000)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 3.3e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.33 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: 0.126 seconds (Warm-up)
Chain 1:                0.088 seconds (Sampling)
Chain 1:                0.214 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: 0.163 seconds (Warm-up)
Chain 2:                0.085 seconds (Sampling)
Chain 2:                0.248 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 1.8e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.18 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: 0.189 seconds (Warm-up)
Chain 3:                0.084 seconds (Sampling)
Chain 3:                0.273 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 1.6e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.16 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: 0.114 seconds (Warm-up)
Chain 4:                0.089 seconds (Sampling)
Chain 4:                0.203 seconds (Total)
Chain 4:
```

```
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    82.29    0.07 1.99 78.63 80.92 82.24 83.64 86.22
beta[1]   5.71    0.07 2.23  1.25  4.20  5.77  7.22  9.95
beta[2]   0.56    0.00 0.06  0.44  0.52  0.56  0.60  0.69
sigma    18.13    0.02 0.61 16.99 17.71 18.10 18.53 19.39
```

```
lp__     -1474.44    0.05 1.46 -1478.16 -1475.16 -1474.09 -1473.36 -1472.67
        n_eff Rhat
alpha     931 1.01
beta[1]   958 1.00
beta[2]  1209 1.00
sigma    1519 1.00
lp__      799 1.00

Samples were drawn using NUTS(diag_e) at Sun Feb 12 22:47:40 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 coefficient estimate is 0.57, this means that every mom's IQ point higher than average the estimated kid score is increased by 0.57.

Question 5

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

```
model5 = lm(kid_score ~ mom_hs+ceneterdiq, data = kidiqdata)
summary(model5)
```

```

Call:
lm(formula = kid_score ~ mom_hs + ceneterdiq, data = kidiqdata)

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 ** 
ceneterdiq   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 
```

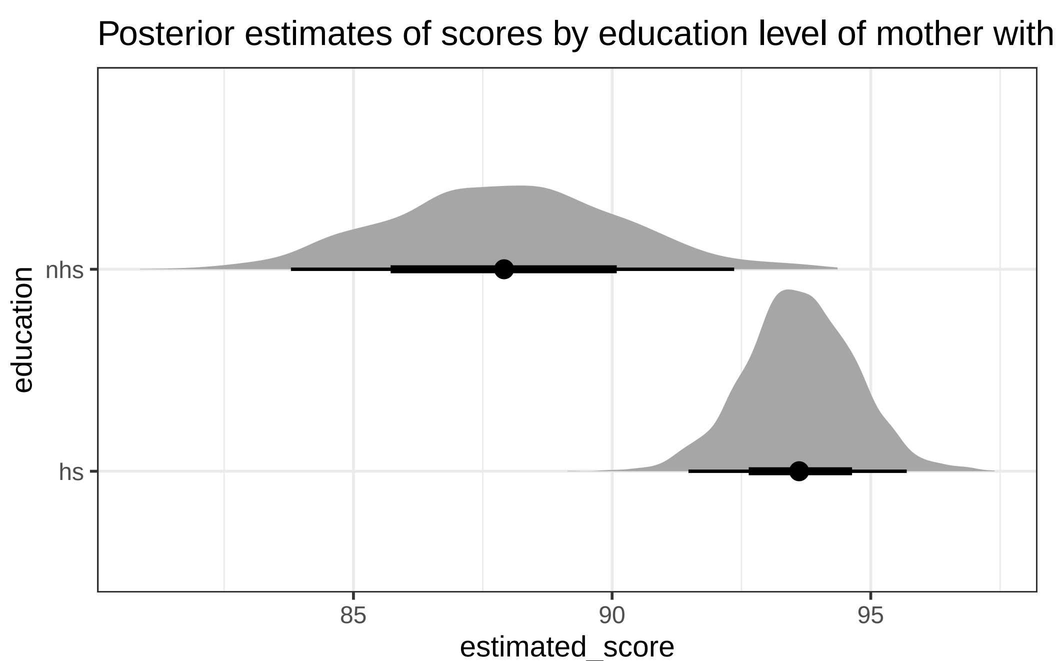
```
F-statistic: 58.72 on 2 and 431 DF,  p-value: < 2.2e-16
```

The result from lm agree with the stan result: in lm we have 82.12, 5.95 and 0.56 for the intercept, beta1, beta2. In Stan, we got 82.31, 5.72 and 0.57

Question 6

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

```
fit3 |>
  spread_draws(alpha, beta[k], sigma) |>
  pivot_wider(names_from = k, names_prefix = "beta", values_from = beta) |>
     mutate(nhs = alpha + beta2*10, # no high school is just the intercept
          hs = alpha + beta1 + beta2*10) |>
  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 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.

```
x_new = 95
post_samples <- extract(fit3)
alpha_hat <- post_samples[["alpha"]]
beta1_hat <- post_samples[["beta"]][,1]
beta2_hat <- post_samples[["beta"]][,2]
sigma_hat <- post_samples[["sigma"]]
lin_pre <- alpha_hat + beta1_hat + -5*beta2_hat
y_new <- rnorm(n = length(sigma_hat), mean = lin_pre, sd = sigma_hat)
hist(y_new)
```