Week 5: Bayesian linear regression and introduction to Stan

11/02/2023

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).

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

The data look like this:

```
kidiq <- read_rds("D:\\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

```
library(skimr)
library(janitor)
library(ggplot2)
skim(kidiq)
```

Table 1: Data summary

Name	kidiq
Number of rows	434
Number of columns	4
Column type frequency:	
numeric	4
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
kid_score	0	1	86.80	20.41	20.00	74.00	90.00	102.00	144.00	
mom_hs	0	1	0.79	0.41	0.00	1.00	1.00	1.00	1.00	
mom_iq	0	1	100.00	15.00	71.04	88.66	97.92	110.27	138.89	
mom_age	0	1	22.79	2.70	17.00	21.00	23.00	25.00	29.00	

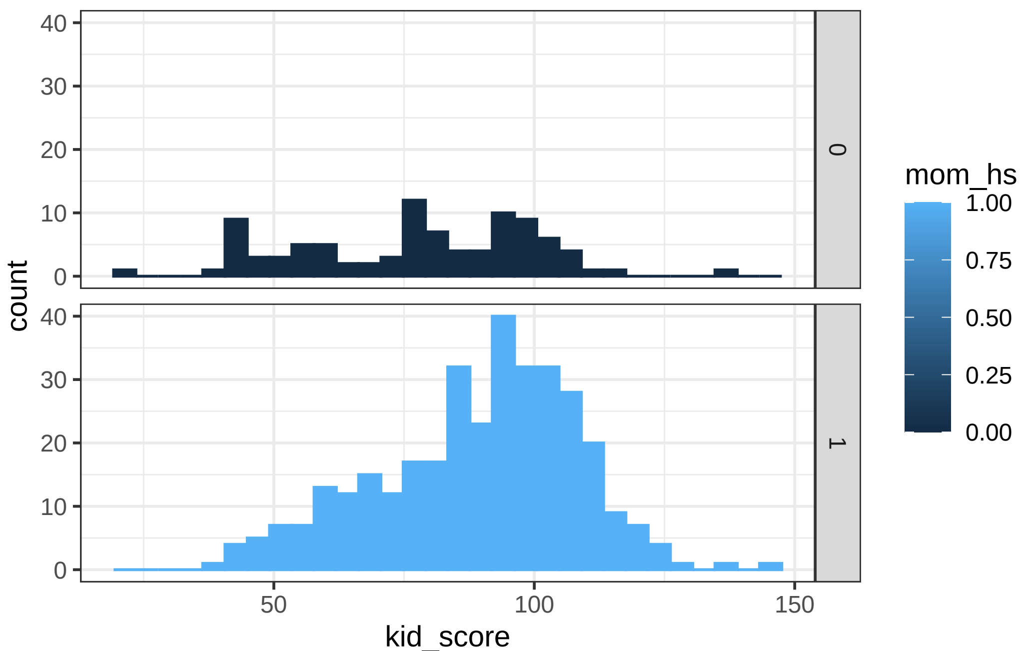
```
kidiq |>get_dupes()
```

```
# A tibble: 2 x 5
  kid_score mom_hs mom_iq mom_age dupe_count
      <int>  <dbl>  <dbl>   <int>      <int>
1       104      1   125.      23          2
2       104      1   125.      23          2
```

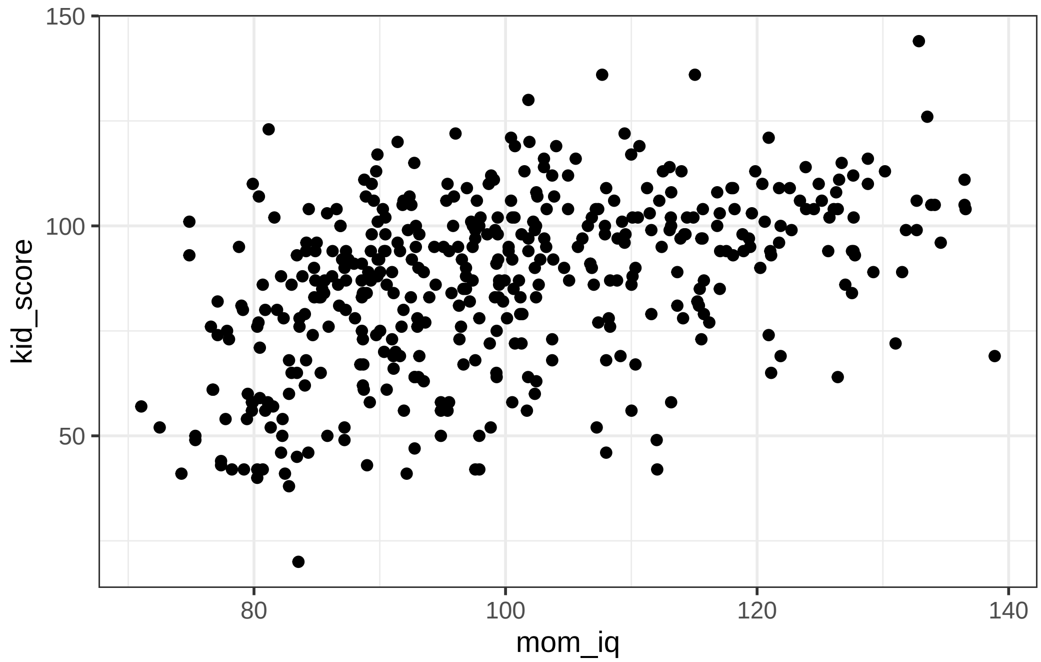
```
kidiq1<-kidiq |> distinct()
summary(kidiq1$mom_age)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  17.00   21.00   23.00   22.79   25.00   29.00 
```

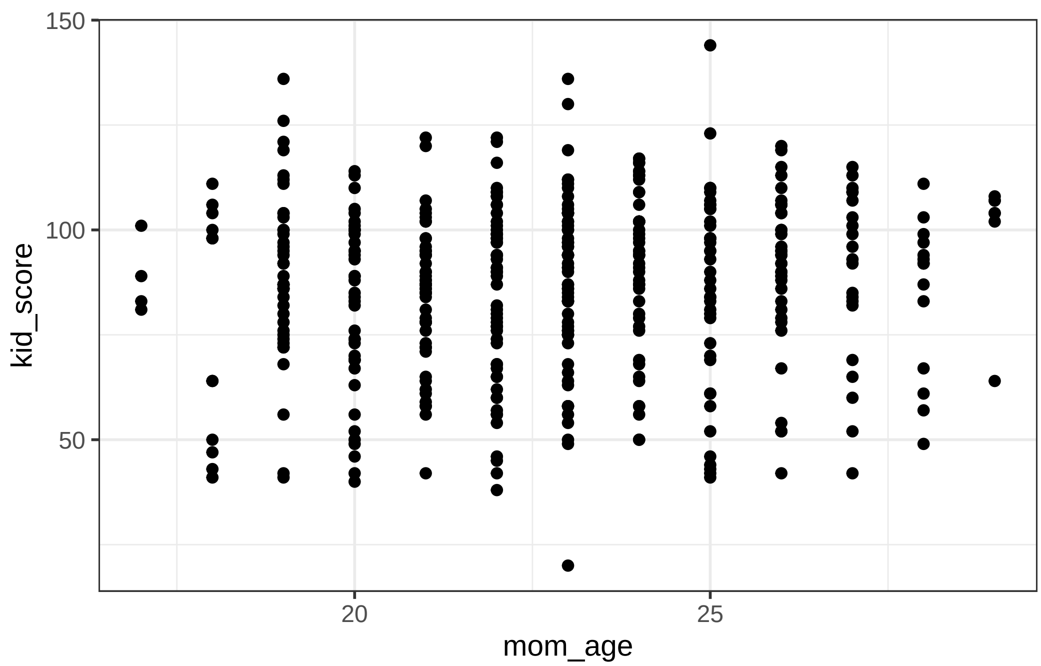
```
kidiq1 |> ggplot(aes(x=kid_score,fill=mom_hs, color=mom_hs)) +geom_histogram( position="id
```




```
kidiq1 |> ggplot()+geom_point(aes(x=mom_iq,y=kid_score))+theme_bw()
```




```
kidiq1 |> ggplot()+geom_point(aes(x=mom_age,y=kid_score))+theme_bw()
```



From the above 3 graphs, the mother who completed high schools education could have a kid with a higher test score.The higher the mother's IQ, the higher her kids's test score. There are no any relationship between kid_score and their mother ages within 30-year-old.

A graph type that's appropriate to the data type is mom_iq VS kid_score plot.

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 <- kidiq1$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:

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

fit <-rstan::stan(file =here::here("D:\\kids2.stan"),
            data = data,
            chains = 3,
            iter = 500)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 3.2e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.32 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: 0.009 seconds (Warm-up)
Chain 1:                0.006 seconds (Sampling)
Chain 1:                0.015 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 6e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.06 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: 0.01 seconds (Warm-up)
Chain 2:                0.005 seconds (Sampling)
```

```
Chain 2:                0.015 seconds (Total)
Chain 2: 

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3: 
Chain 3: Gradient evaluation took 5e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.05 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: 0.01 seconds (Warm-up)
Chain 3:                0.005 seconds (Sampling)
Chain 3:                0.015 seconds (Total)
Chain 3: 
```

Look at the summary

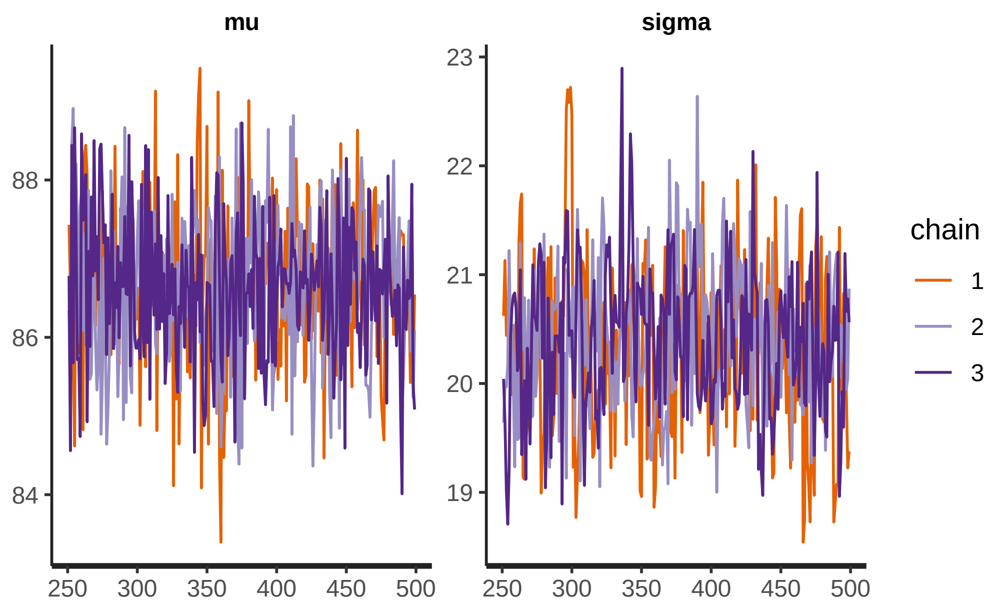
```
fit
```

```
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       86.65    0.03 0.96    84.69    85.99    86.67    87.31    88.48   887
sigma    20.37    0.03 0.70    19.05    19.88    20.38    20.83    21.72   407
lp__  -1522.39    0.06 1.00 -1525.08 -1522.71 -1522.09 -1521.69 -1521.40   268
      Rhat
mu    1.00
```

```
sigma 1.00
lp__  1.02

Samples were drawn using NUTS(diag_e) at Thu Feb  9 14:11:10 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).
```

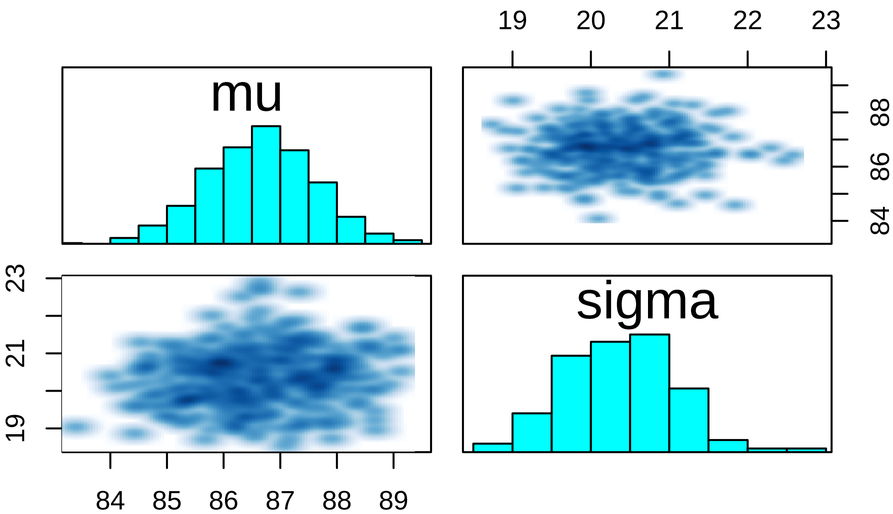
Traceplot

```r
traceplot(fit)
```

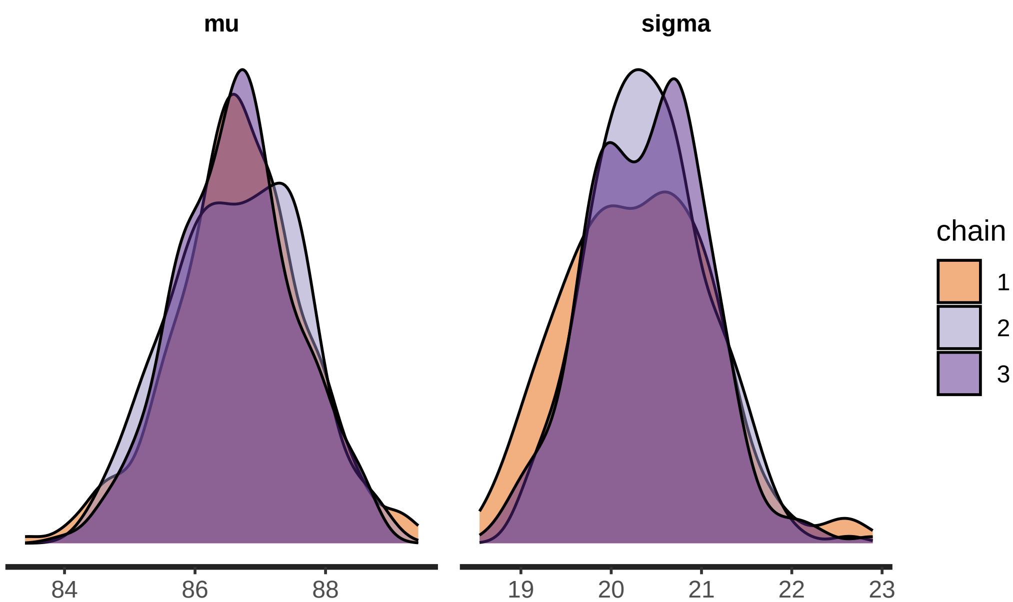


All looks fine.

```r
library(tidyverse)
library(rstan)
library(tidybayes)
library(here)
pairs(fit, pars = c("mu", "sigma"))
```




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



Understanding output

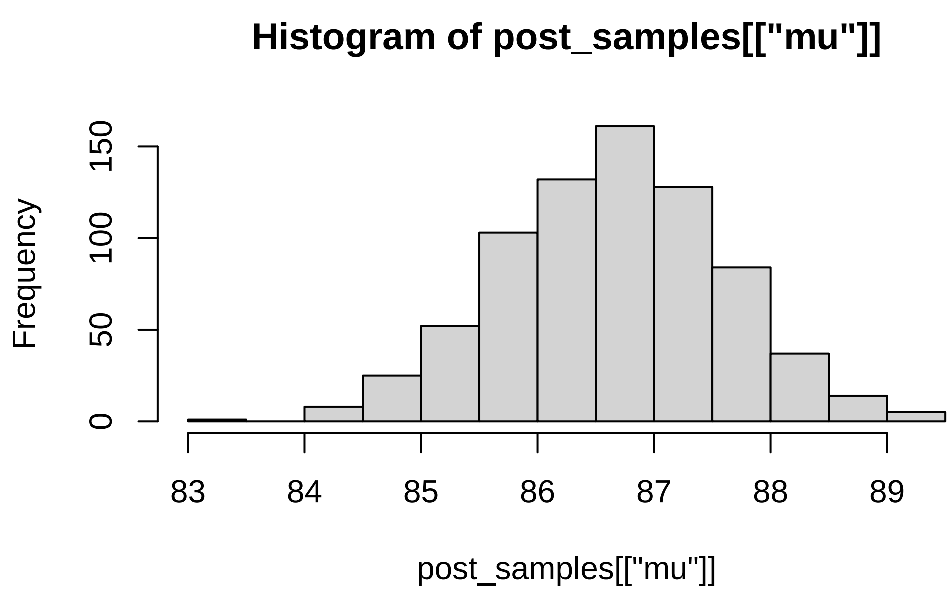
What does the model actually give us? A number of samples from the posteriors. To see this, we can use `extract` to get the samples.

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

```
[1] 86.67280 89.09272 86.82312 85.88996 85.79614 86.09840
```

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] 86.66638
```

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

```
    2.5%
84.68704
```

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

```
   97.5%
88.47602
```

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          87.4
 2      1          2     2 mu          86.6
 3      1          3     3 mu          85.8
 4      1          4     4 mu          85.7
 5      1          5     5 mu          84.6
 6      1          6     6 mu          87.9
 7      1          7     7 mu          87.2
 8      1          8     8 mu          86.6
 9      1          9     9 mu          85.8
10      1         10    10 mu          87.7
# ... with 1,490 more rows
```

```r
# 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  87.4  20.6
 2      1          2     2  86.6  21.1
 3      1          3     3  85.8  20.4
 4      1          4     4  85.7  20.8
 5      1          5     5  84.6  20.5
 6      1          6     6  87.9  20.1
 7      1          7     7  87.2  19.9
 8      1          8     8  86.6  20.5
 9      1          9     9  85.8  20.0
10      1         10    10  87.7  20.1
# ... with 740 more rows
```

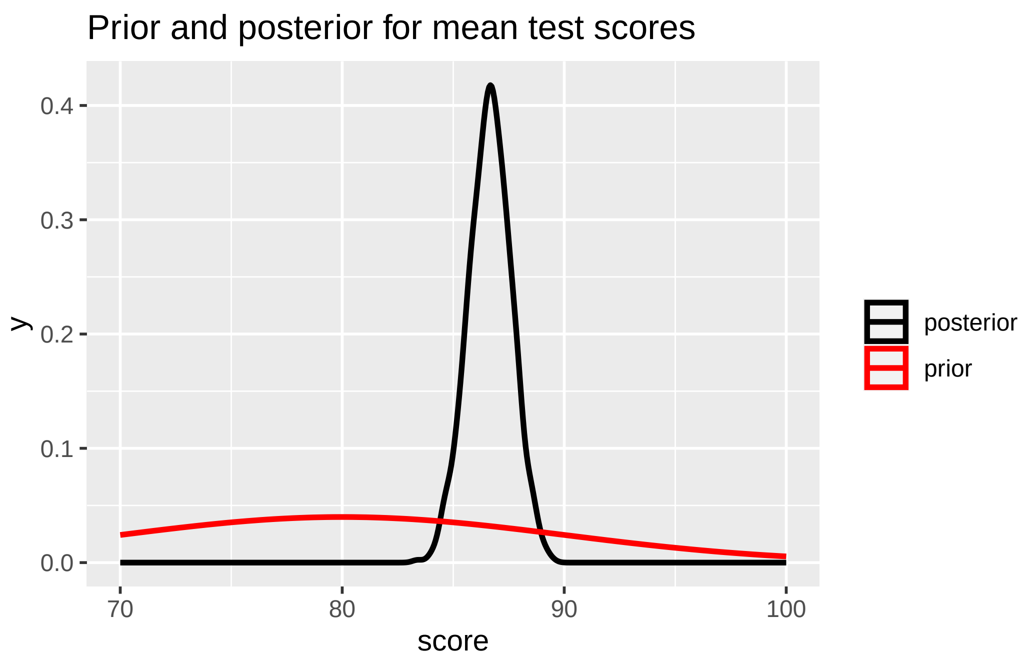
```r
# quickly calculate the quantiles using

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   85.4   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

```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.

```
mod1 <- rstan::stan(file = here::here("D:\\kids4.stan"),
            data = data,
            chains = 3,
            iter = 500)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 4.1e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.41 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: 0.012 seconds (Warm-up)
Chain 1:                0.011 seconds (Sampling)
Chain 1:                0.023 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 8e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:   1 / 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: 0.017 seconds (Warm-up)
Chain 2:                0.011 seconds (Sampling)
Chain 2:                0.028 seconds (Total)
Chain 2:
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 8e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 3: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 3: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 3: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 3: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 3: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 3: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 3: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 3: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 3: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 3: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 3: Iteration: 500 / 500 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.012 seconds (Warm-up)
Chain 3:                0.011 seconds (Sampling)
Chain 3:                0.023 seconds (Total)
Chain 3:
```

```r
summary(mod1)
```

```
$summary
             mean     se_mean         sd         2.5%          25%         50%
mu      86.744247 0.012220141 0.31529066    86.126758    86.533919    86.73682
sigma    6.354886 0.003882331 0.05163474     6.246828     6.320766     6.35618
lp__  -5049.097933 0.079255276 1.07939971 -5051.998742 -5049.487676 -5048.78689
              75%        97.5%    n_eff      Rhat
mu      86.970020    87.350719 665.6866 0.9982755
sigma    6.389746     6.457376 176.8882 1.0266805
lp__  -5048.340445 -5048.034356 185.4848 1.0198484

$c_summary
, , chains = chain:1

         stats
```

```
parameter          mean          sd          2.5%          25%          50%
    mu        86.761912 0.31889034    86.179366    86.556805    86.786076
    sigma      6.358401 0.05136026     6.259532     6.318483     6.361942
    lp__   -5049.108748 0.96347001 -5051.727325 -5049.487676 -5048.863589
         stats
parameter           75%        97.5%
    mu        86.999158    87.315121
    sigma      6.394533     6.458287
    lp__   -5048.399305 -5048.059445

, , chains = chain:2

         stats
parameter          mean          sd          2.5%          25%          50%
    mu        86.724840 0.29651214    86.108713    86.536745    86.717945
    sigma      6.347547 0.04412865     6.248212     6.320625     6.350895
    lp__   -5048.887725 0.93036517 -5051.421288 -5049.217430 -5048.597489
         stats
parameter           75%        97.5%
    mu        86.924553    87.340447
    sigma      6.375844     6.427458
    lp__   -5048.249221 -5048.021918

, , chains = chain:3

         stats
parameter          mean          sd          2.5%          25%          50%
    mu        86.745991 0.32972424    86.146077    86.510753    86.71888
    sigma      6.358709 0.05790921     6.240723     6.324686     6.36040
    lp__   -5049.297327 1.27530995 -5052.731377 -5049.870413 -5048.91278
         stats
parameter           75%        97.5%
    mu        86.978554    87.366738
    sigma      6.394654     6.467573
    lp__   -5048.390519 -5048.024420
```

mu did not change,but sigma totally were altered and became smaller.

```
mod1samples <- mod1  |>
  gather_draws(mu, sigma) # gather = long format
mod1samples
```

```
# A tibble: 1,500 x 5
# Groups:   .variable [2]
   .chain .iteration .draw .variable .value
    <int>      <int> <int> <chr>      <dbl>
 1      1          1     1 mu          86.6
 2      1          2     2 mu          87.2
 3      1          3     3 mu          86.8
 4      1          4     4 mu          86.8
 5      1          5     5 mu          86.9
 6      1          6     6 mu          87.2
 7      1          7     7 mu          87.1
 8      1          8     8 mu          86.7
 9      1          9     9 mu          86.9
10      1         10    10 mu          86.6
# ... with 1,490 more rows
```

```
# wide format
mod1  |>  spread_draws(mu, sigma)
```

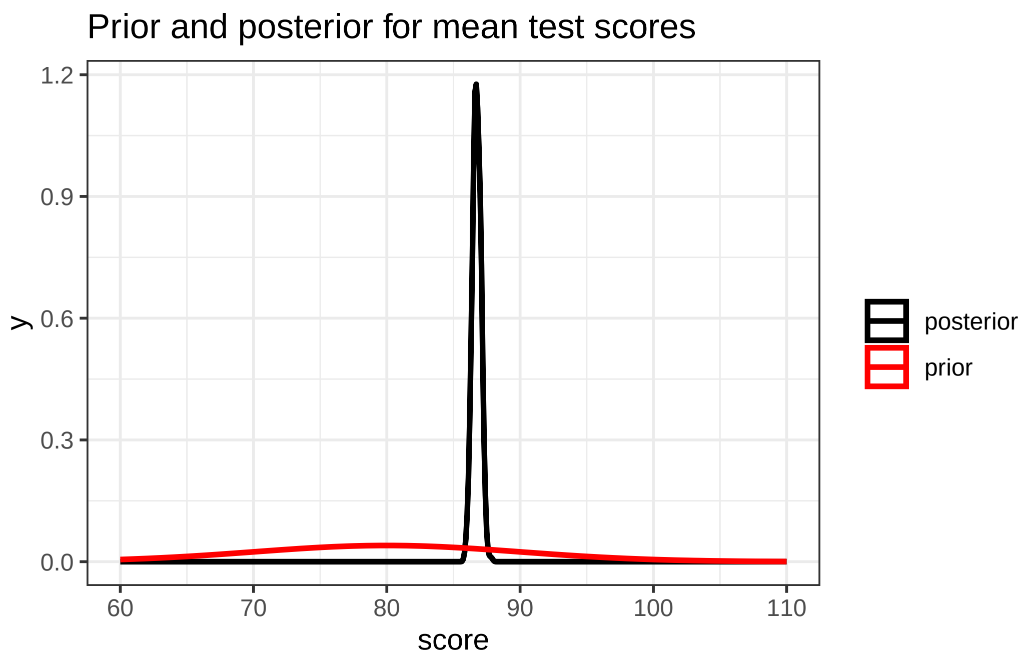
```
# A tibble: 750 x 5
   .chain .iteration .draw    mu sigma
    <int>      <int> <int> <dbl> <dbl>
 1      1          1     1  86.6  6.31
 2      1          2     2  87.2  6.30
 3      1          3     3  86.8  6.33
 4      1          4     4  86.8  6.36
 5      1          5     5  86.9  6.40
 6      1          6     6  87.2  6.42
 7      1          7     7  87.1  6.38
 8      1          8     8  86.7  6.39
 9      1          9     9  86.9  6.30
10      1         10    10  86.6  6.30
# ... with 740 more rows
```

```
# quickly calculate the quantiles using

mod1samples |>
  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   86.3   87.1     0.8 median qi
2 sigma       6.36   6.29   6.42    0.8 median qi
```

```
mod1samples |>
  filter(.variable == "mu") |>
  ggplot(aes(.value, color = "posterior")) + geom_density(size = 1) +
  xlim(c(60, 110)) +
  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") + theme_bw()
```



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(kidiq1$mom_hs, ncol = 1) # force this to be a matrix
K <- 1

data <- list(y = y, N = length(y),
             X =X, K = K)
fit2 <- rstan::stan(file = here::here("D:\\kids3.stan"),
            data = data,
            iter = 1000)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 7.6e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.76 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: 0.174 seconds (Warm-up)
Chain 1:                0.081 seconds (Sampling)
Chain 1:                0.255 seconds (Total)
Chain 1:
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 2.1e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.21 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: 0.15 seconds (Warm-up)
Chain 2:                0.079 seconds (Sampling)
Chain 2:                0.229 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 2.1e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.21 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: 0.227 seconds (Warm-up)
Chain 3:                0.096 seconds (Sampling)
Chain 3:                0.323 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 2.1e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.21 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: 0.136 seconds (Warm-up)
Chain 4:                0.089 seconds (Sampling)
Chain 4:                0.225 seconds (Total)
Chain 4:
```

Question 3

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

```r
library(skimr)
library(janitor)
```

```r
mod2<-lm(kid_score~mom_hs,data=kidiq1)
summary(fit2)$summary[c("alpha", "beta[1]"),]
```

```
            mean    se_mean       sd      2.5%       25%      50%      75%
alpha   77.94506 0.06481637 1.972842 74.195136 76.615324 77.86691 79.29679
beta[1] 11.20894 0.07466591 2.219853  6.922637  9.714296 11.22368 12.68583
           97.5%    n_eff     Rhat
alpha   81.71793 926.4355 1.003990
beta[1] 15.77321 883.9012 1.002249
```

```r
summary(mod2)
```

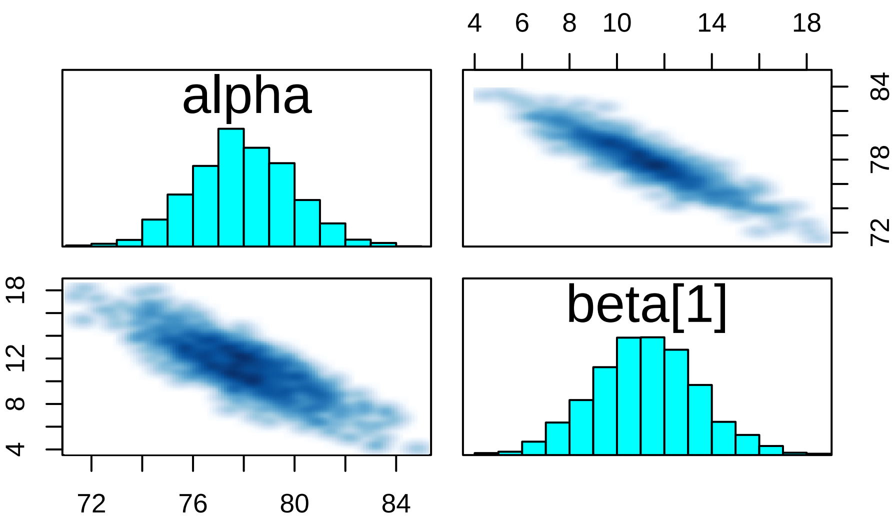
```

Call:
lm(formula = kid_score ~ mom_hs, data = kidiq1)

Residuals:
    Min      1Q  Median      3Q     Max 
-57.548 -13.276   2.724  14.724  58.452 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   77.548      2.060  37.651  < 2e-16 ***
mom_hs        11.728      2.324   5.046 6.67e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 19.86 on 431 degrees of freedom
Multiple R-squared:  0.05578,   Adjusted R-squared:  0.05358 
F-statistic: 25.46 on 1 and 431 DF,  p-value: 6.675e-07
```

Both are almost same.

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

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



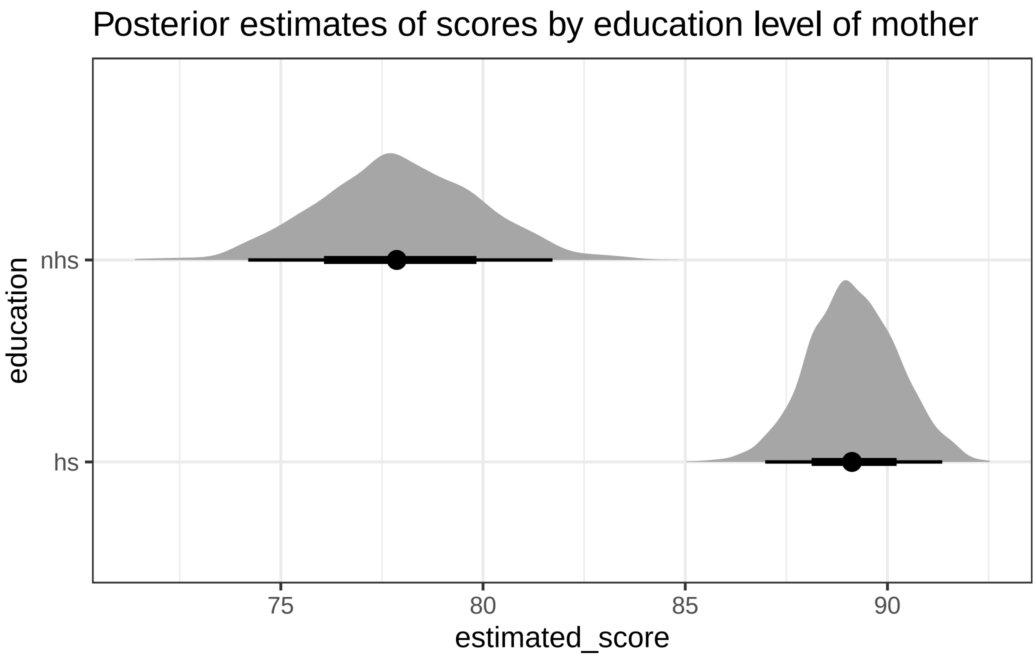
In the fit2, its intercept(alpha) has a wide distribution which means a good sampling but a little hard to compute the intercept and beta(slope) has a narrower distribution which means a bad sampling but easily to compute the slope.Thus,this is potentially a problem.

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.

```
y <- kidiq1$kid_score
mu0 <- 80
sigma0 <- 10
# named list to input for stan function
data <- list(y = y,
             N = length(y),
             mu0 = mu0,
             sigma0 = sigma0)

X <-  cbind(as.matrix(kidiq1$mom_hs),as.matrix(kidiq1$mom_iq - mean(kidiq1$mom_iq))) # for
K <- 2
```

```r
data <- list(y = y, N = length(y),
             X =X, K = K)
mod3 <- stan(file = here::here("D:\\kids3.stan"),
            data = data,
            iter = 1000)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 2e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.2 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:                0.101 seconds (Sampling)
Chain 1:                0.227 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.9e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.19 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: 0.168 seconds (Warm-up)
Chain 2:                0.096 seconds (Sampling)
Chain 2:                0.264 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: 0.142 seconds (Warm-up)
Chain 3:                0.082 seconds (Sampling)
Chain 3:                0.224 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
```

```
Chain 4: Gradient evaluation took 1.9e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.19 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: 0.145 seconds (Warm-up)
Chain 4:                0.09 seconds (Sampling)
Chain 4:                0.235 seconds (Total)
Chain 4:
```

```r
summary(mod3)$summary[c("alpha", "beta[1]","beta[2]"),]
```

```
             mean     se_mean         sd       2.5%        25%        50%
alpha   82.2959454 0.058773590 1.84507402 78.7395384 81.0690597 82.2719577
beta[1]  5.6793521 0.066466361 2.10914158  1.6091746  4.2796599  5.7035114
beta[2]  0.5633618 0.001679147 0.05979105  0.4474414  0.5249119  0.5629014
              75%      97.5%     n_eff      Rhat
alpha   83.5103386 85.9420213  985.5148 1.0023039
beta[1]  7.1565947  9.6547576 1006.9494 1.0011341
beta[2]  0.6030604  0.6788208 1267.9296 0.9994368
```

```r
mean(kidiq1$kid_score)
```

```
[1] 86.75751
```

Here the alpha(intercept) means that when the mom_iq was in the average of all mom_iqs, the kids'test score actually was what should be.It almost closes to the mean of kid_score.The intercept was changed from the non-centered data before.

Here the beta[1] is an estimator that shows a positive relationship between the kids'test score and moms' education.It means that when the mom completed the high school education their kids'test score also increased 5.6837998 scores corresponding to the moms' education variation.

Here the beta[2] is an estimator that shows a positive relationship between the kids'test score and moms' IQ.It means that when the moms' IQ changed in one unit and their kids'test score also altered 0.5656852 scores corresponding to the moms' IQ variation.

Question 5

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

```
library(tidyverse)
library(stringr)
library(dplyr)
library(janitor)

kidiq2<- kidiq1 |> mutate(mom_iq=mom_iq-mean(mom_iq))
kidiq3<-as.data.frame(kidiq2)

mod4<- lm(kid_score~mom_hs+mom_iq,data=kidiq3)
summary(mod4)$coeff
```

```
              Estimate Std. Error   t value      Pr(>|t|)
(Intercept) 82.0859652 1.94540681 42.194756 5.838177e-155
mom_hs       5.9493443 2.21435766  2.686713  7.495551e-03
mom_iq       0.5633781 0.06081298  9.264110  9.493682e-19
```

The 3 estimators of both formulas are almost same.

Question 6

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

```
library(plyr)
library(dplyr)
library(tidyverse)

post_mod3_samples <- extract(mod3)
```

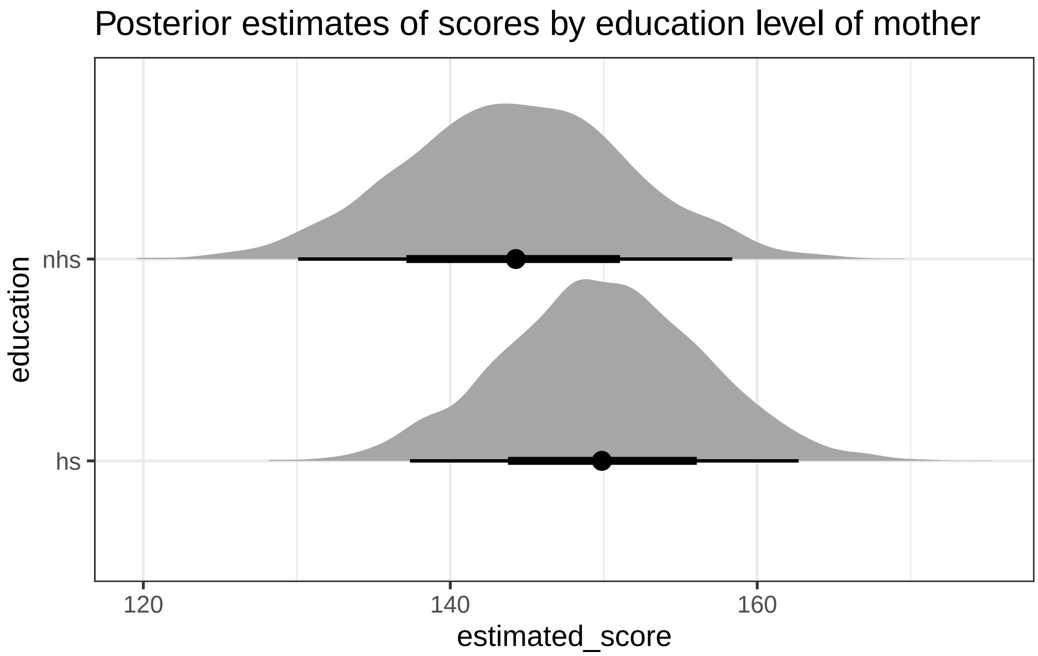
```r
length(post_mod3_samples)
```

```
[1] 4
```

```r
dim(post_mod3_samples[["beta"]])
```

```
[1] 2000    2
```

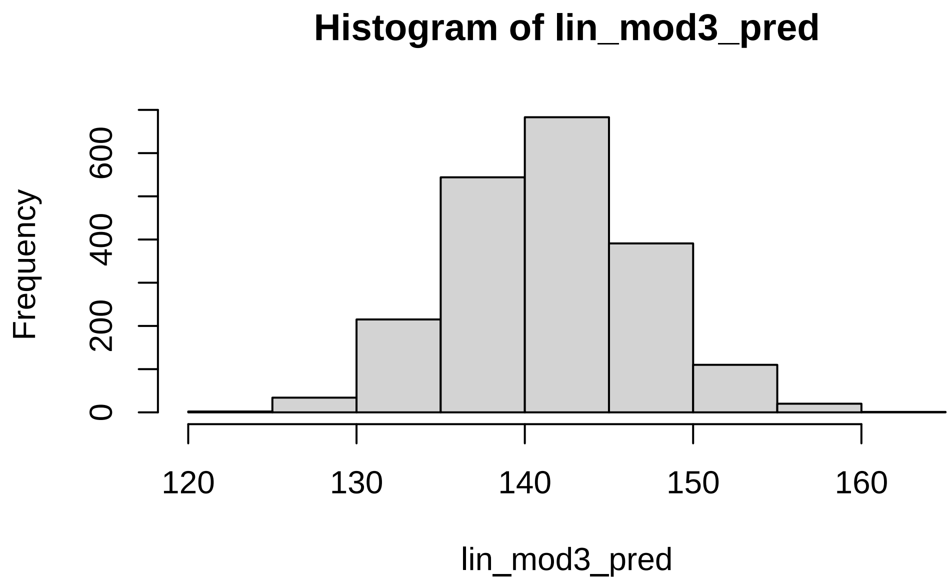
```r
x_new1 <- 110

mod3 |>
  spread_draws(alpha, beta[k]) |> pivot_wider( names_from = "k", values_from = "beta")|>
  dplyr::rename(beta1="1",beta2="2")|>
     mutate(nhs = alpha+beta2*x_new1, # no high school is just the intercept
          hs = alpha + beta1+beta2*x_new1) |>
  select(nhs, hs) |>
  pivot_longer(nhs:hs, names_to = "education", values_to = "estimated_score") |>
  #plyr::ddply("education", summarise, grp.mean=mean(estimated_score)) |>
  ggplot(aes(y = education, x = estimated_score)) +
  stat_halfeye()+
  theme_bw() +
  ggtitle("Posterior estimates of scores by education level of mother")
```



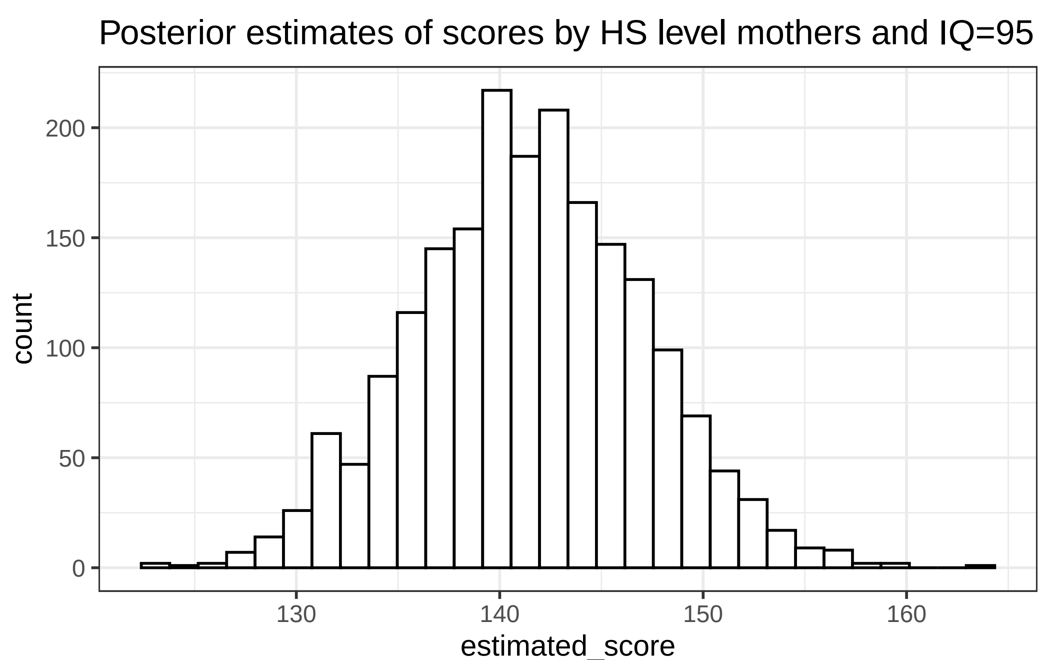
Question 7

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.

```
library(ggplot2)
mod3_alpha <- post_mod3_samples[["alpha"]]
mod3_beta1 <- post_mod3_samples[["beta"]][,1]
mod3_beta2 <- post_mod3_samples[["beta"]][,2]
x_new2 <-95
lin_mod3_pred <- mod3_alpha + mod3_beta1*1+mod3_beta2*x_new2
hist(lin_mod3_pred)
```




```r
as.data.frame(lin_mod3_pred) |> ggplot(aes(x=lin_mod3_pred))+geom_histogram(color="black",
```



===