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

```r
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	

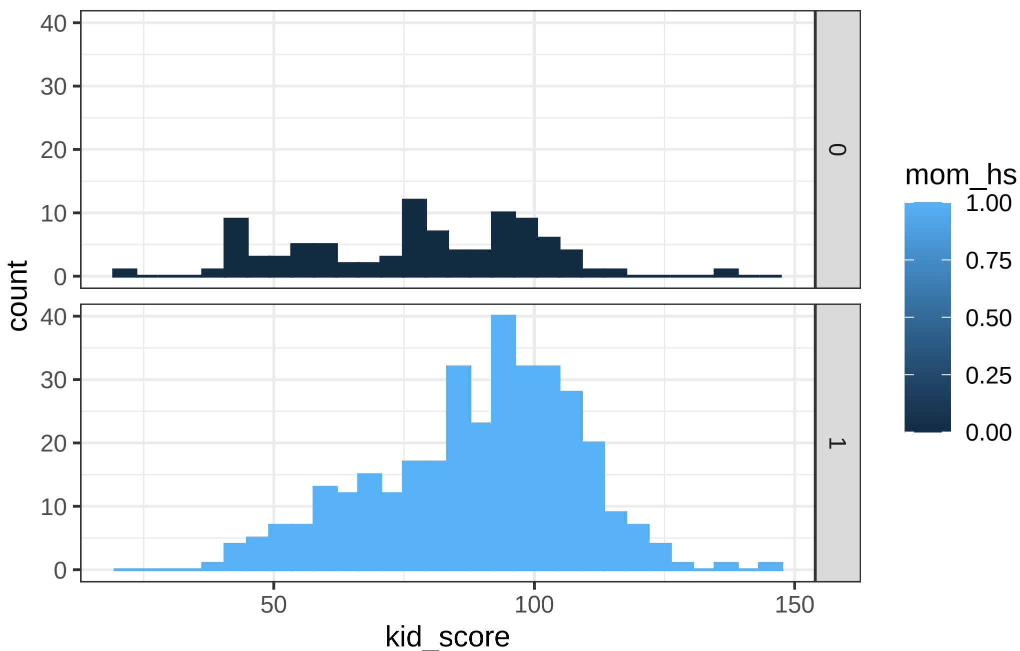
```r
kidiq |>get_dupes()
```

```
# 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
```

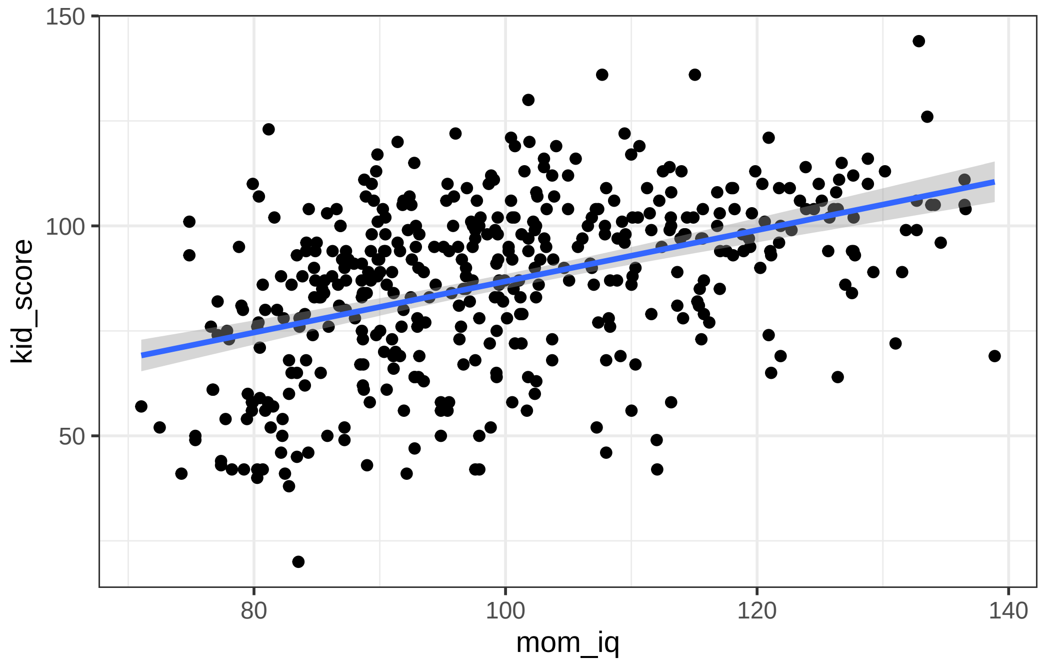
```r
kidiq1<-kidiq |> distinct()
summary(kidiq1$mom_age)
```

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

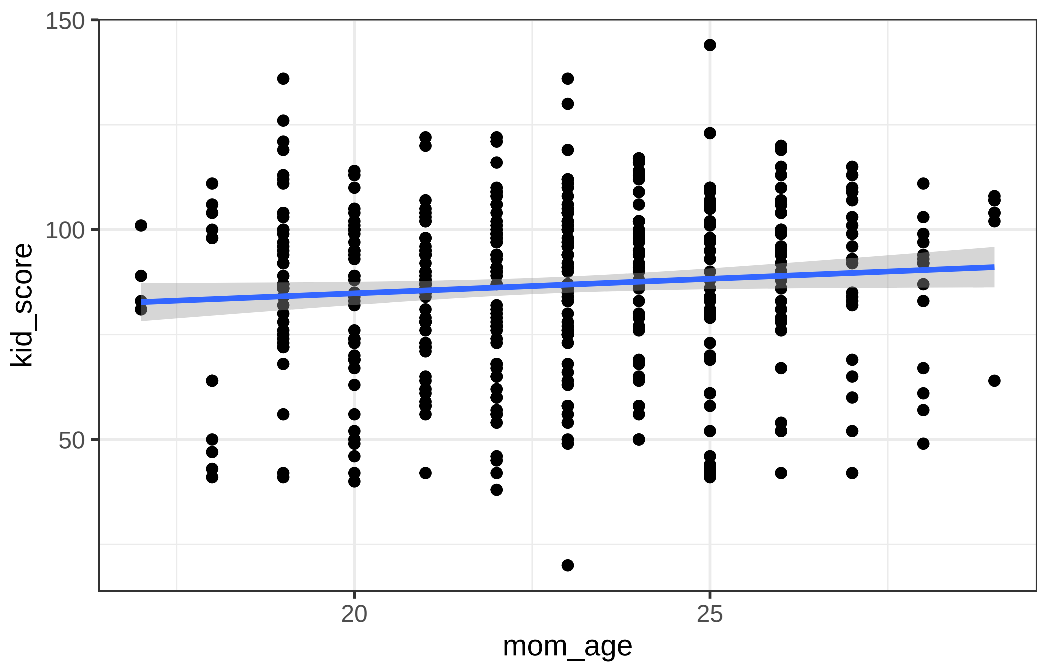
```r
kidiq1 |> ggplot(aes(x=kid_score,fill=mom_hs, color=mom_hs)) +geom_histogram( position="id
```




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




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



From the above 3 graphs, the mother who completed high schools education could have kids with a higher test score.

The higher the mother's IQ, the higher her kids' test score. The measurement should conform to our prior knowledge from our basic gene genetics .

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 5.8e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.58 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: 0.013 seconds (Warm-up)
Chain 1:                0.007 seconds (Sampling)
Chain 1:                0.02 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 7e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.07 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: 0.013 seconds (Warm-up)
Chain 2:                0.008 seconds (Sampling)
Chain 2:                0.021 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: 0.013 seconds (Warm-up)
Chain 3:                0.009 seconds (Sampling)
Chain 3:                0.022 seconds (Total)
Chain 3:
```

Look at the summary

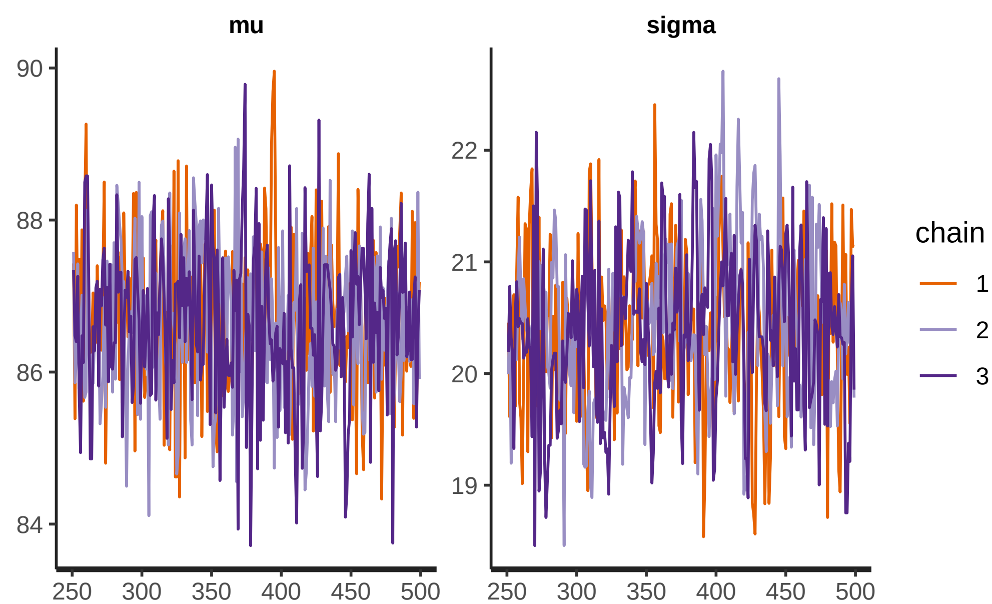
```
fit
```

```
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       86.67    0.03 0.99    84.71    86.01    86.66    87.32    88.53   810
sigma    20.39    0.04 0.72    18.95    19.93    20.37    20.87    21.77   334
```

```
lp__   -1522.45    0.05 1.00 -1525.26 -1522.93 -1522.13 -1521.69 -1521.41   399
      Rhat
mu       1
sigma    1
lp__     1

Samples were drawn using NUTS(diag_e) at Sat Feb 11 17:19:57 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

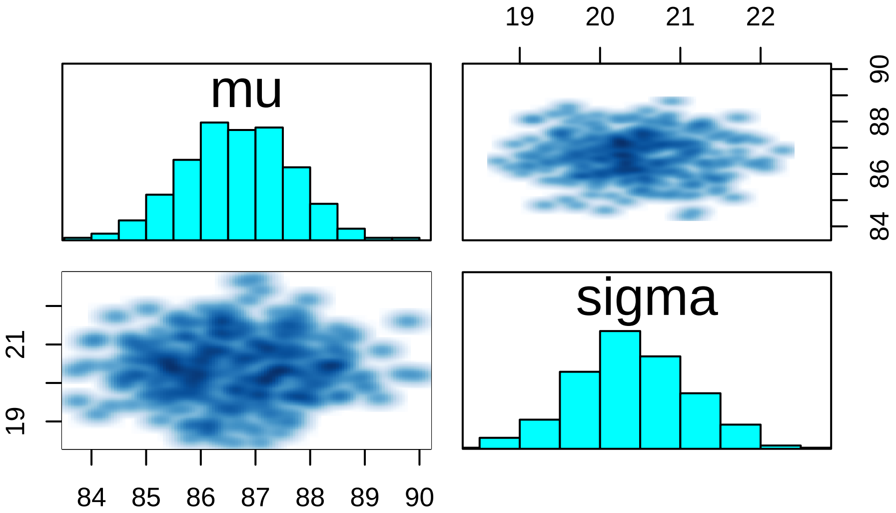
```
traceplot(fit)
```



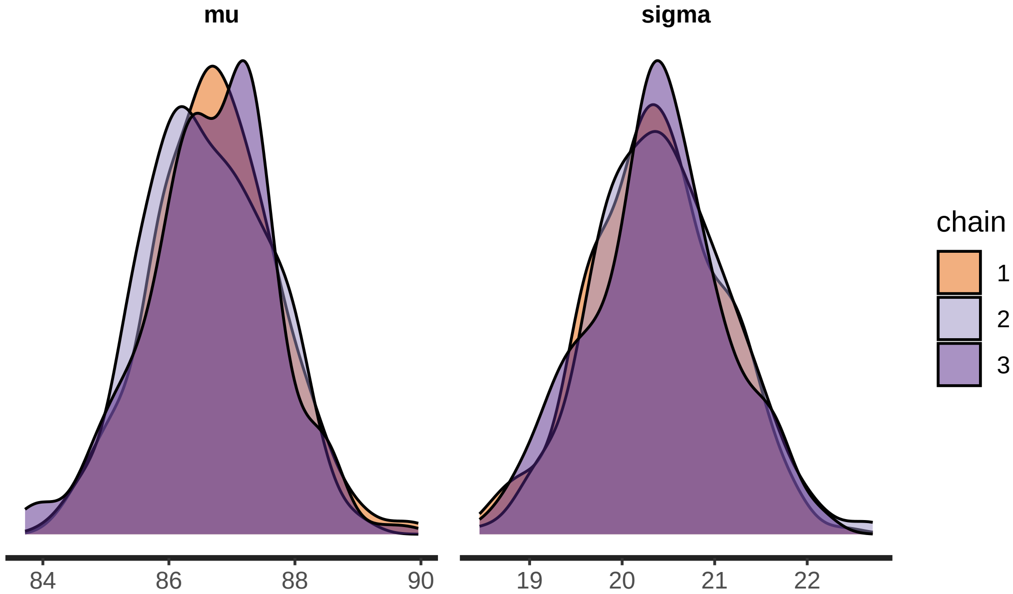
All looks fine.

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

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




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



Understanding output

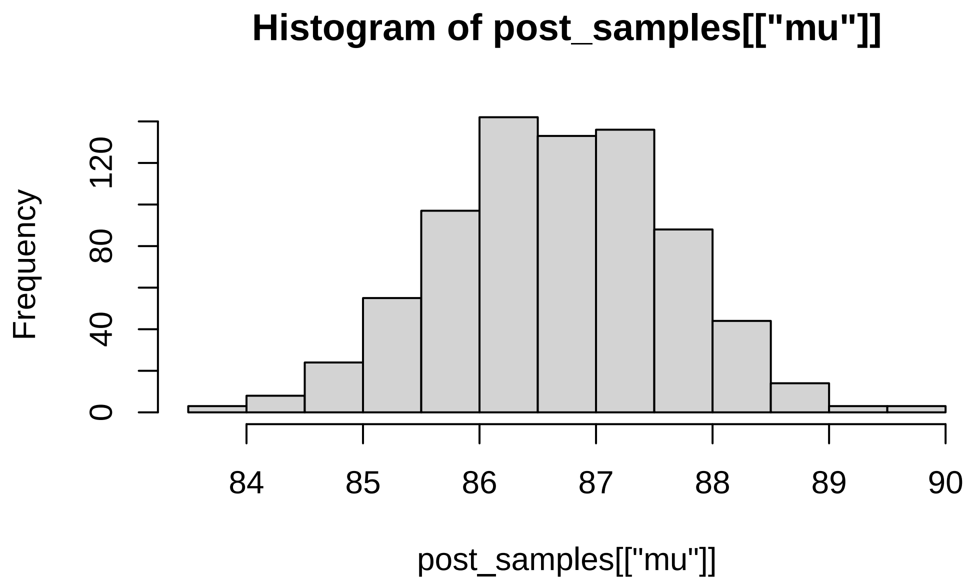
What does the model actually give us? A number of samples from the posteriors. To see this, we can use `extract` to get the samples.

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

```
[1] 87.04942 86.07437 86.52241 88.34940 86.44363 85.86745
```

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] 86.66447
```

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

```
    2.5% 
84.71028 
```

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

```
   97.5% 
88.53152 
```

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          85.4
 3      1          3     3 mu          88.2
 4      1          4     4 mu          86.7
 5      1          5     5 mu          87.5
 6      1          6     6 mu          85.7
 7      1          7     7 mu          87.9
 8      1          8     8 mu          85.6
 9      1          9     9 mu          88.2
10      1         10    10 mu          89.3
# ... 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  87.4  20.5
 2      1          2     2  85.4  19.6
 3      1          3     3  88.2  20.4
 4      1          4     4  86.7  20.2
 5      1          5     5  87.5  20.7
```

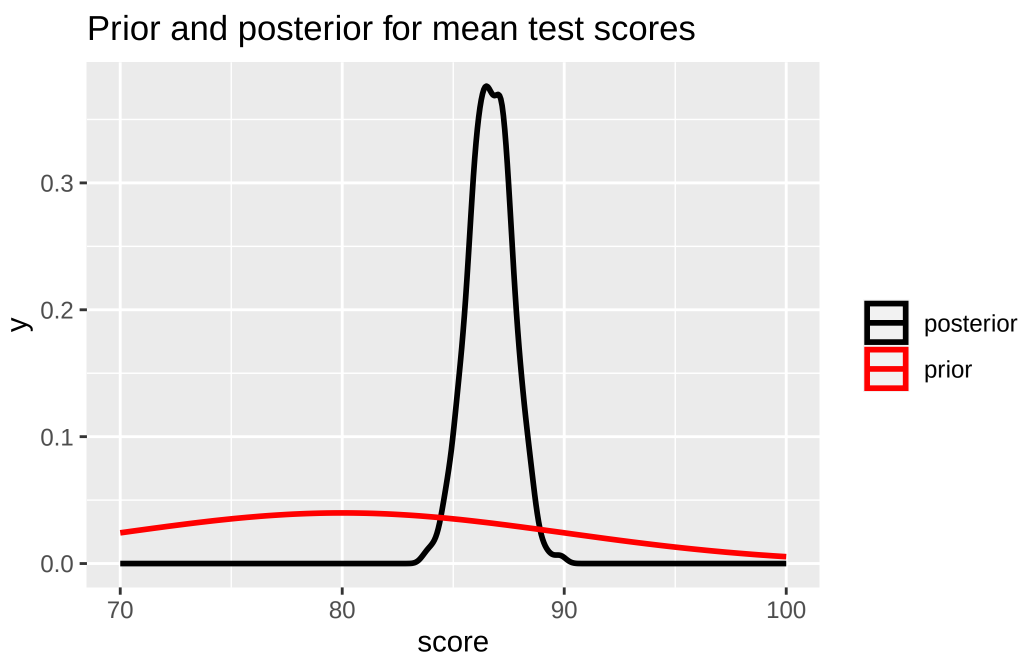
```
 6       1           6     6  85.7  19.7
 7       1           7     7  87.9  20.9
 8       1           8     8  85.6  21.6
 9       1           9     9  88.2  19.8
10       1          10    10  89.3  19.6
# ... with 740 more rows
```

```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   21.3    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.

```
sigma0=0.1
data1 <- list(y = y,
             N = length(y),
             mu0 = mu0,
             sigma0 = sigma0)

mod1 <- rstan::stan(file = here::here("D:\\kids2.stan"),
            data = data1,
            chains = 3,
            iter = 2000)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 6e-06 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
Chain 1: Adjust your expectations accordingly!
```

```
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.02 seconds (Warm-up)
Chain 1:                0.017 seconds (Sampling)
Chain 1:                0.037 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 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.019 seconds (Warm-up)
Chain 2:                0.018 seconds (Sampling)
Chain 2:                0.037 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 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.02 seconds (Warm-up)
Chain 3:                0.019 seconds (Sampling)
Chain 3:                0.039 seconds (Total)
Chain 3:
```

```r
summary(mod1)
```

```
$summary
            mean    se_mean         sd        2.5%         25%         50%
mu      80.06239 0.00177281 0.09852525    79.86668    79.99531    80.06244
sigma   21.42284 0.01356464 0.70544159    20.13811    20.93251    21.39048
lp__  -1544.64491 0.02520868 0.95410573 -1547.28710 -1545.03151 -1544.34725
             75%       97.5%    n_eff      Rhat
mu      80.12937    80.25518 3088.664 0.9991818
sigma   21.86678    22.87857 2704.616 1.0001777
lp__  -1543.96521 -1543.70523 1432.494 1.0028459

$c_summary
, , chains = chain:1
```

```
         stats
parameter        mean        sd        2.5%         25%         50%         75%
    mu       80.06235 0.1007538    79.85893    79.99529    80.06589    80.12688
    sigma    21.41244 0.7399982    19.96335    20.88934    21.38038    21.90048
    lp__  -1544.71662 1.0364538 -1547.52793 -1545.12767 -1544.38886 -1543.98575
         stats
parameter       97.5%
    mu       80.26373
    sigma    22.94567
    lp__  -1543.71239

, , chains = chain:2

         stats
parameter        mean         sd        2.5%         25%         50%
    mu       80.06107 0.09984337    79.86668    79.99094    80.06118
    sigma    21.42073 0.70635145    20.09671    20.93677    21.40125
    lp__  -1544.65950 0.97578501 -1547.36930 -1545.03795 -1544.32499
         stats
parameter         75%       97.5%
    mu       80.13397    80.25953
    sigma    21.84670    22.85757
    lp__  -1543.97139 -1543.70288

, , chains = chain:3

         stats
parameter        mean         sd        2.5%         25%         50%
    mu       80.06374 0.09496054    79.87364    80.00006    80.06251
    sigma    21.43534 0.66868597    20.25224    20.97234    21.38505
    lp__  -1544.55861 0.83280754 -1546.76699 -1544.93648 -1544.32068
         stats
parameter         75%       97.5%
    mu       80.12715    80.24499
    sigma    21.87611    22.86054
    lp__  -1543.93865 -1543.70722
```

mu was greatly changed from 86.67 to the current value 80.06593,but sigma was smally altered from 20.40 to the current value 21.37953.

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

```
# A tibble: 6,000 x 5
# Groups:   .variable [2]
   .chain .iteration .draw .variable .value
    <int>      <int> <int> <chr>      <dbl>
 1      1          1     1 mu          80.2
 2      1          2     2 mu          80.1
 3      1          3     3 mu          80.0
 4      1          4     4 mu          80.2
 5      1          5     5 mu          80.0
 6      1          6     6 mu          80.2
 7      1          7     7 mu          80.1
 8      1          8     8 mu          80.1
 9      1          9     9 mu          79.9
10      1         10    10 mu          80.1
# ... with 5,990 more rows
```

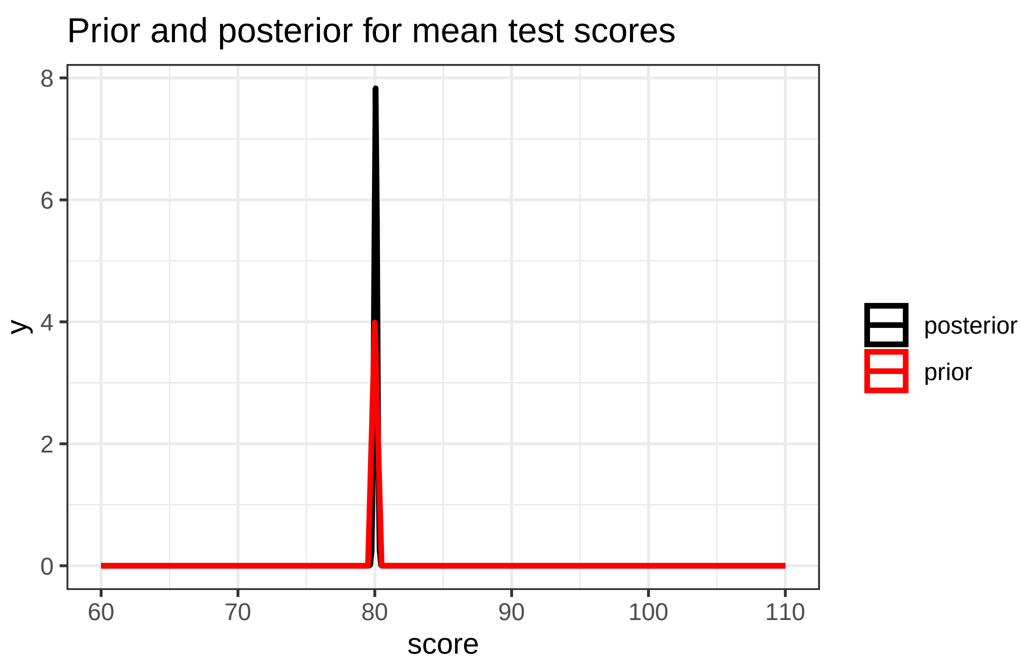
```r
# wide format
mod1  |>  spread_draws(mu, sigma)
```

```
# A tibble: 3,000 x 5
   .chain .iteration .draw    mu sigma
    <int>      <int> <int> <dbl> <dbl>
 1      1          1     1  80.2  21.1
 2      1          2     2  80.1  20.2
 3      1          3     3  80.0  20.6
 4      1          4     4  80.2  21.9
 5      1          5     5  80.0  20.7
 6      1          6     6  80.2  21.4
 7      1          7     7  80.1  22.1
 8      1          8     8  80.1  22.6
 9      1          9     9  79.9  20.6
10      1         10    10  80.1  22.4
# ... with 2,990 more rows
```

```r
# 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          80.1   79.9   80.2    0.8 median qi
2 sigma       21.4   20.5   22.3    0.8 median qi
```

```r
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 6.9e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.69 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: 0.128 seconds (Warm-up)
Chain 1:                0.072 seconds (Sampling)
Chain 1:                0.2 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 2e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.2 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: 0.153 seconds (Warm-up)
Chain 2:                0.1 seconds (Sampling)
Chain 2:                0.253 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: 0.171 seconds (Warm-up)
Chain 3:                0.105 seconds (Sampling)
Chain 3:                0.276 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 2.2e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.22 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: 0.143 seconds (Warm-up)
Chain 4:                0.09 seconds (Sampling)
Chain 4:                0.233 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)

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   77.95438 0.07733880 1.982904 74.13960 76.657763 77.94530 79.24086
beta[1] 11.17409 0.08799721 2.248487  6.92756  9.706822 11.23988 12.70730
           97.5%    n_eff     Rhat
alpha   81.86837 657.3679 1.002831
beta[1] 15.60107 652.8946 1.003038
```

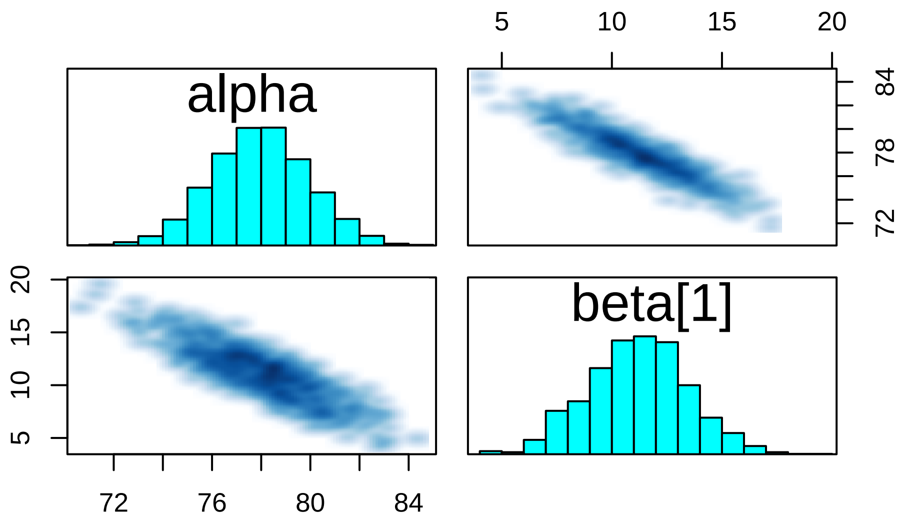
```r
summary(mod2)
```

```
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
```

So the STAN results are mean=77.98760 and beta[1]=11.14227,while the LM results are mean=77.548 and beta[1]=11.728.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"))
```

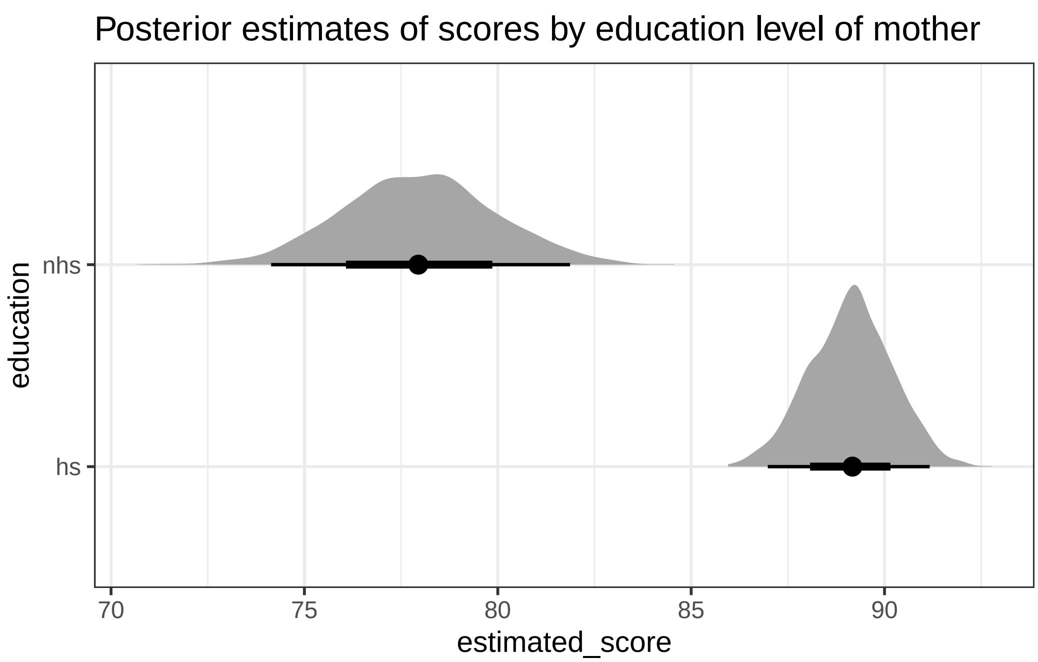


In the fit2, its intercept(alpha) has a wide distribution which means a good sampling but a little hard to compute the intercept and beta(slope) has a narrower distribution which means a bad sampling but easily to compute the slope.Thus,this is potentially a problem.

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
```

```r
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

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 3.5e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.35 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: 0.21 seconds (Warm-up)
Chain 1:                0.122 seconds (Sampling)
Chain 1:                0.332 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: 0.198 seconds (Warm-up)
Chain 2:                0.105 seconds (Sampling)
Chain 2:                0.303 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 2.2e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.22 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: 0.235 seconds (Warm-up)
Chain 3:                0.13 seconds (Sampling)
Chain 3:                0.365 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 2.3e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.23 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: 0.181 seconds (Warm-up)
Chain 4:                0.113 seconds (Sampling)
Chain 4:                0.294 seconds (Total)
Chain 4:
```

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

```
              mean     se_mean         sd       2.5%        25%        50%
alpha   82.2803912 0.063258008 1.95122124 78.5624594 80.9113344 82.2603408
beta[1]  5.7020846 0.072405791 2.20687393  1.3214906  4.2434353  5.7589902
beta[2]  0.5634241 0.001563907 0.05923281  0.4500216  0.5223463  0.5644201
               75%      97.5%     n_eff     Rhat
alpha   83.5348608 86.2339818  951.4413 1.000797
beta[1]  7.2795312  9.8474624  928.9844 1.001632
beta[2]  0.6045248  0.6764668 1434.5065 1.000366
```

```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 and moms did not complete their high school education, the kids' test score actually was what should be(82.217).The centered intercept should be totally different 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 level.It means that when the moms completed their high school education her kids' test score also increased by 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 increased or reduced by one unit and their kids' test score also increased or reduced by 0.5656852 scores corresponding to the moms' IQ variation.

Question 5

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

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


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

```
              mean     se_mean         sd       2.5%        25%        50%
alpha   82.2803912 0.063258008 1.95122124 78.5624594 80.9113344 82.2603408
beta[1]  5.7020846 0.072405791 2.20687393  1.3214906  4.2434353  5.7589902
beta[2]  0.5634241 0.001563907 0.05923281  0.4500216  0.5223463  0.5644201
               75%      97.5%     n_eff      Rhat
alpha   83.5348608 86.2339818  951.4413 1.000797
beta[1]  7.2795312  9.8474624  928.9844 1.001632
beta[2]  0.6045248  0.6764668 1434.5065 1.000366
```

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

mod4<- lm(kid_score~mom_hs+mom_iq,data=kidiq2)
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 in both models are almost same.

Question 6

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

```r
library(plyr)
library(dplyr)
library(tidyverse)
mean_mom_iq<-mean(kidiq1$mom_iq)
mean_mom_iq
```

```
[1] 99.94338
```

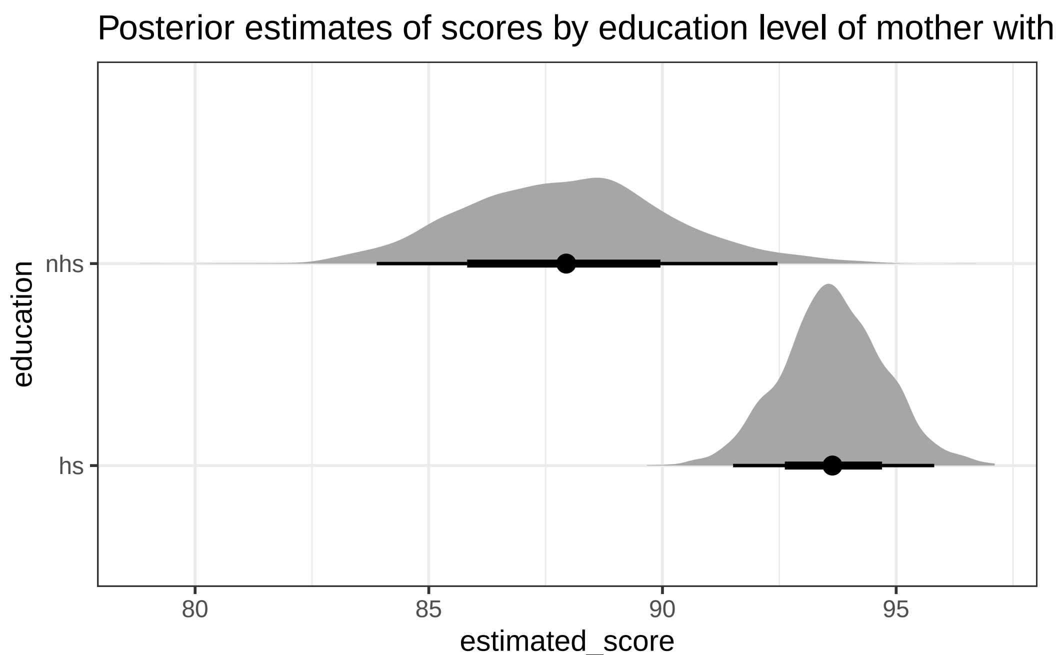
```r
post_mod3_samples <- extract(mod3)
length(post_mod3_samples)
```

```
[1] 4
```

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

```
[1] 2000    2
```

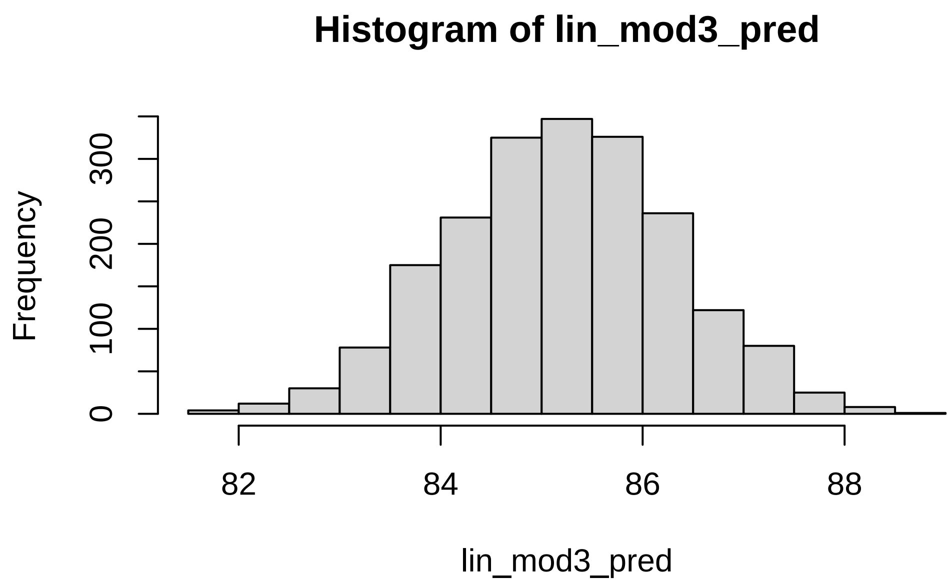
```
x_new1 <- 110-mean_mom_iq

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 with IQ=110")
```



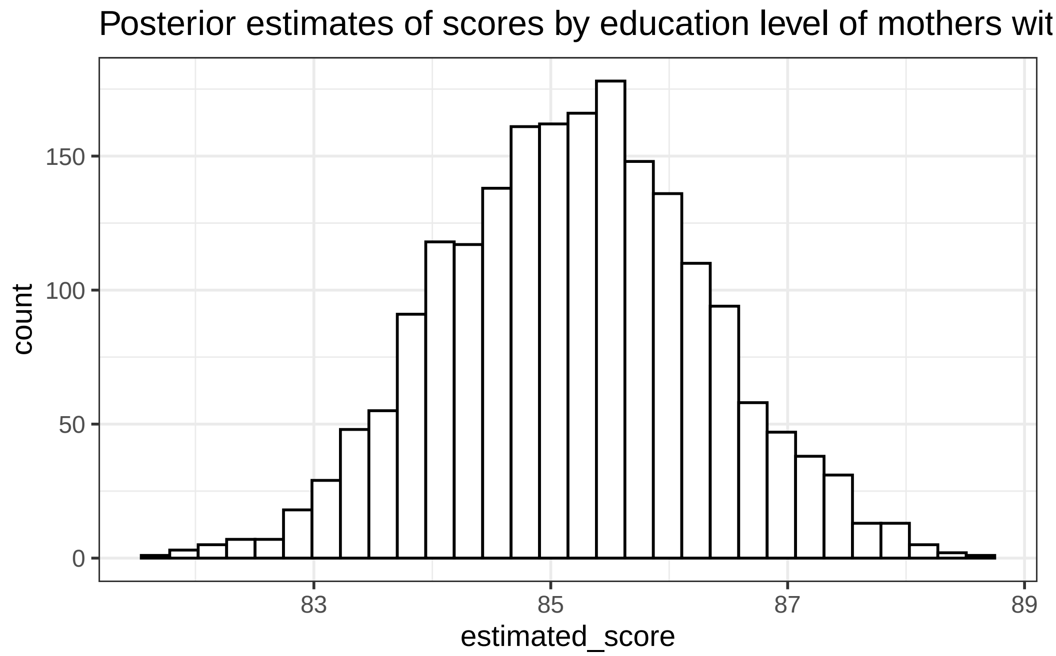
Question 7

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.

```
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-mean_mom_iq
lin_mod3_pred <- mod3_alpha + mod3_beta1*1+mod3_beta2*x_new2
hist(lin_mod3_pred)
```




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


Posterior estimates of scores by education level of mothers wit
count
150
100
50
0
83
85
87
89
estimated_score

===