

Week 5: Bayesian linear regression and introduction to Stan

07/02/24

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
# i 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

Estimating mean, no covariates

In class we were trying to estimate the mean and standard deviation of the kid's test scores. The `kids2.stan` file contains a Stan model to do this. If you look at it, you will notice the first `data` chunk lists some inputs that we have to define: the outcome variable `y`, number of observations `N`, and the mean and standard deviation of the prior on `mu`. Let's define all these values in a `data` list.

```
y <- kidiq$kid_score
mu0 <- 80
sigma0 <- 10

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

Now we can run the model:

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

```

Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHeaders/include/stan/math/
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Core/
namespace Eigen {
~

/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Core/
namespace Eigen {
~
;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHeaders/include/stan/math/
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/Core:
#include <complex>
~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

```

```

SAMPLING FOR MODEL 'kids2' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 1.6e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.16 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: 0.011192 seconds (Warm-up)

```

Chain 1: 0.004972 seconds (Sampling)
Chain 1: 0.016164 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'kids2' 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: 0.011851 seconds (Warm-up)
Chain 2: 0.004201 seconds (Sampling)
Chain 2: 0.016052 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'kids2' 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: 0.00798 seconds (Warm-up)
Chain 3:           0.005107 seconds (Sampling)
Chain 3:           0.013087 seconds (Total)
Chain 3:

```

Look at the summary

```
fit
```

Inference for Stan model: kids2.

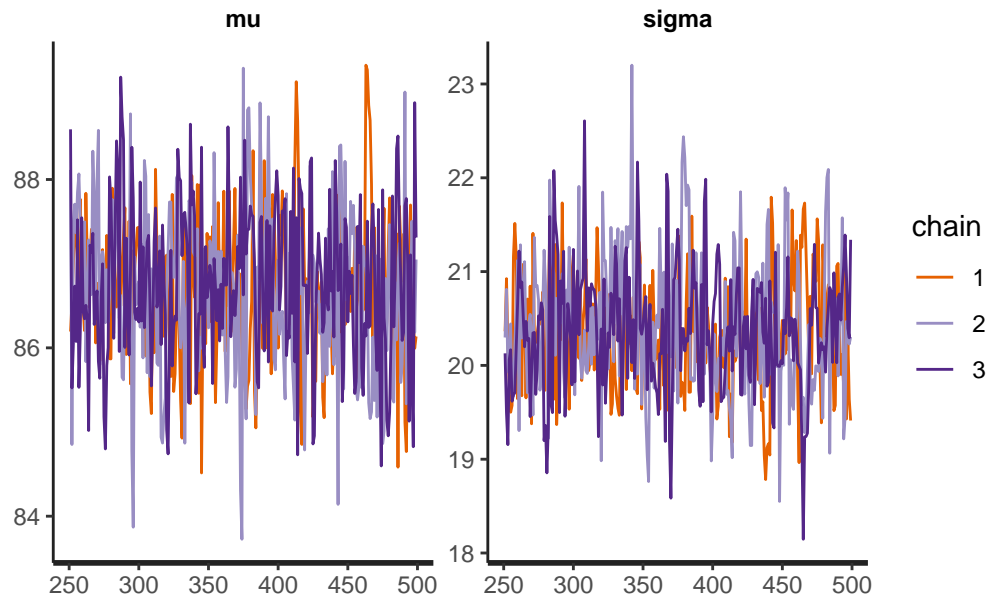
3 chains, each with iter=500; warmup=250; thin=1;
post-warmup draws per chain=250, total post-warmup draws=750.

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff
mu	86.73	0.04	0.91	84.87	86.13	86.74	87.35	88.59	515
sigma	20.38	0.03	0.67	19.17	19.94	20.36	20.80	21.84	370
lp__	-1525.66	0.06	0.96	-1528.47	-1525.97	-1525.37	-1525.00	-1524.78	283
Rhat									
mu	1.00								
sigma	1.00								
lp__	1.02								

Samples were drawn using NUTS(diag_e) at Wed Feb 7 09:03:14 2024.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).

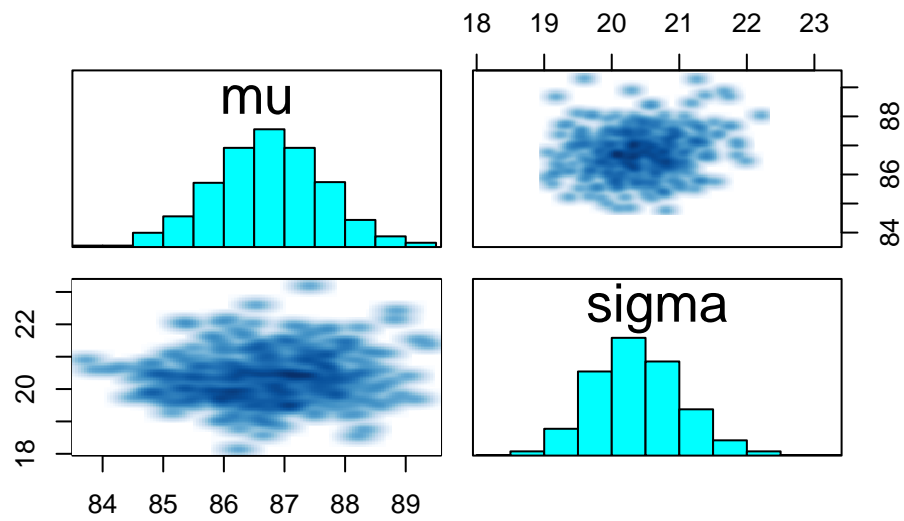
Traceplot

```
traceplot(fit)
```

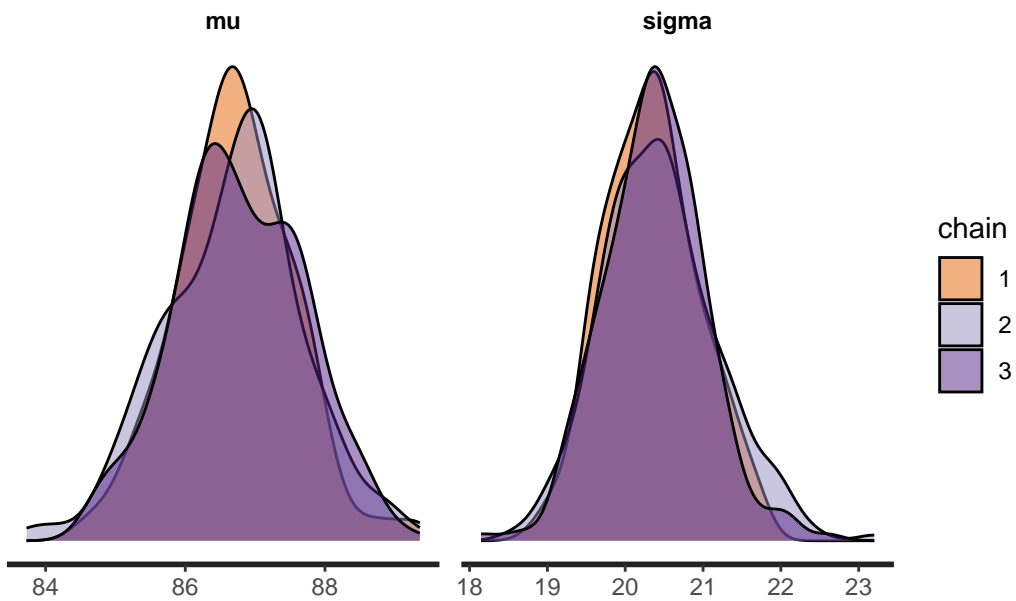


All looks fine.

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



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



Understanding output

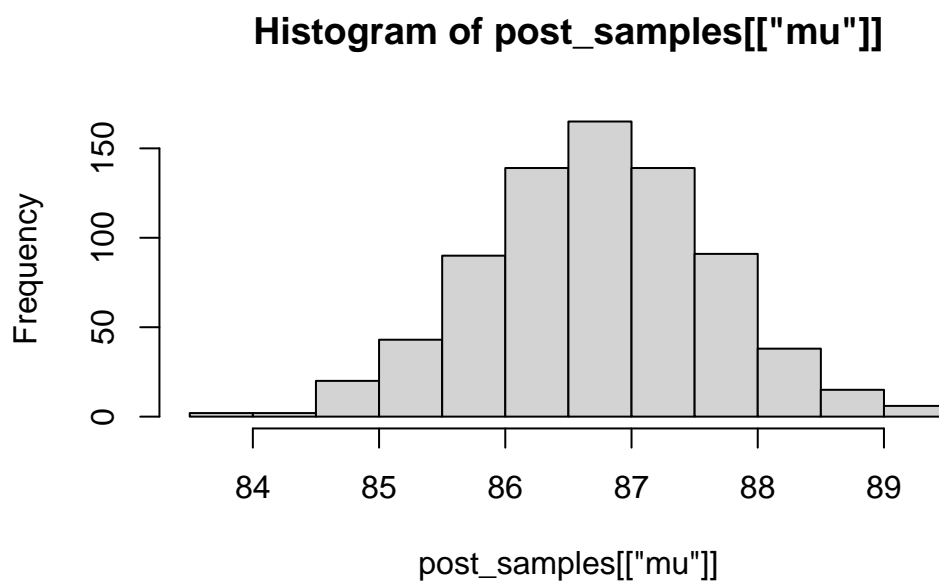
What does the model actually give us? A number of samples from the posteriors. To see this, we can use `extract` to get the samples.

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

```
[1] 85.61213 86.62474 87.48899 86.65172 86.96016 86.31551
```

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] 86.73793
```



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

```
2.5%
84.86876
```

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

```
97.5%
88.587
```

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         86.2
2       1         2     2 mu         86.5
3       1         3     3 mu         87.0
4       1         4     4 mu         87.6
5       1         5     5 mu         87.3
6       1         6     6 mu         87.0
7       1         7     7 mu         86.3
8       1         8     8 mu         87.8
9       1         9     9 mu         86.9
10      1        10    10 mu         86.5
# i 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  86.2  20.4
2       1         2     2  86.5  20.9
3       1         3     3  87.0  20.2
4       1         4     4  87.6  19.8
5       1         5     5  87.3  19.5
6       1         6     6  87.0  19.6
7       1         7     7  86.3  21.2
8       1         8     8  87.8  21.5
9       1         9     9  86.9  21.2
10      1        10    10  86.5  19.6
# i 740 more rows
```

```
# quickly calculate the quantiles using
```

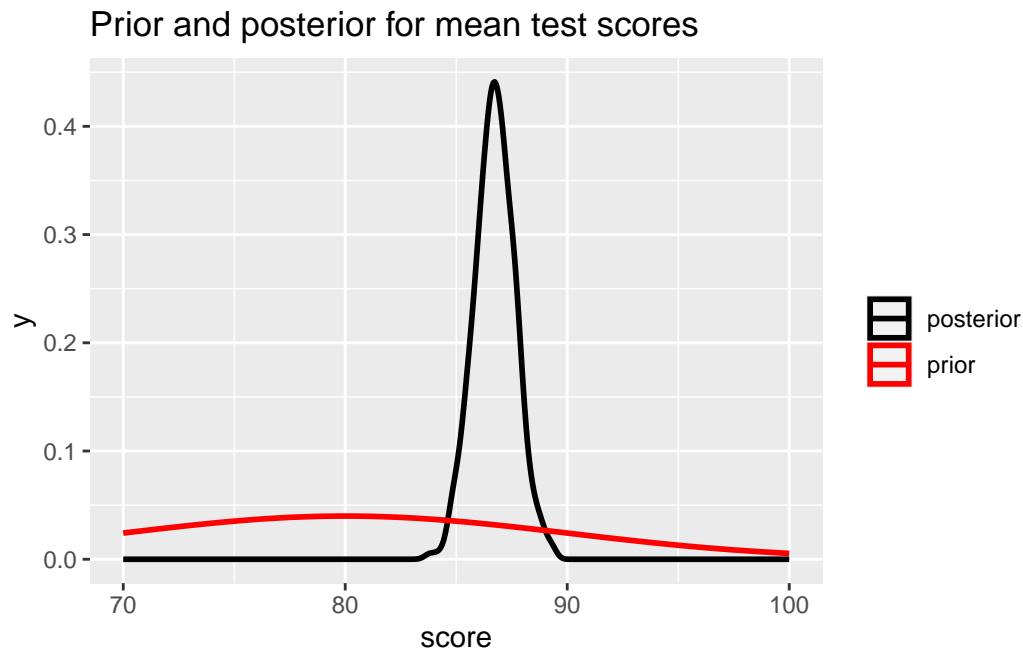
```
dsamples |>
  median_qi(.width = 0.8)
```

```
# A tibble: 2 x 7
  .variable .value .lower .upper .width .point .interval
  <chr>      <dbl> <dbl> <dbl> <dbl> <chr> <chr>
1 mu        86.7  85.5  87.8   0.8 median qi
2 sigma     20.4  19.6  21.3   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.

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("code/models/kids3.stan"),
             data = data,
             iter = 1000)

```

```

Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHeaders/include/StanHeaders:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/Core:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Core/
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Core/
namespace Eigen {
~
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Core/
namespace Eigen {
~
~
;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHeaders/include/StanHeaders:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/Core:
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/Core:
#include <complex>
~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

```

```

SAMPLING FOR MODEL 'kids3' 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 / 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: 0.132816 seconds (Warm-up)
Chain 1: 0.058024 seconds (Sampling)
Chain 1: 0.19084 seconds (Total)
Chain 1:

```

SAMPLING FOR MODEL 'kids3' 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: 0.130526 seconds (Warm-up)
Chain 2: 0.080514 seconds (Sampling)
Chain 2: 0.21104 seconds (Total)
Chain 2:

```

SAMPLING FOR MODEL 'kids3' NOW (CHAIN 3).

```

Chain 3:
Chain 3: Gradient evaluation took 1.7e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.17 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: 0.1357 seconds (Warm-up)
Chain 3:                  0.078099 seconds (Sampling)
Chain 3:                  0.213799 seconds (Total)
Chain 3:

```

SAMPLING FOR MODEL 'kids3' NOW (CHAIN 4).

```

Chain 4:
Chain 4: Gradient evaluation took 1.8e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.18 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: 0.144232 seconds (Warm-up)
Chain 4:                  0.082369 seconds (Sampling)

```

Chain 4: 0.226601 seconds (Total)
Chain 4:

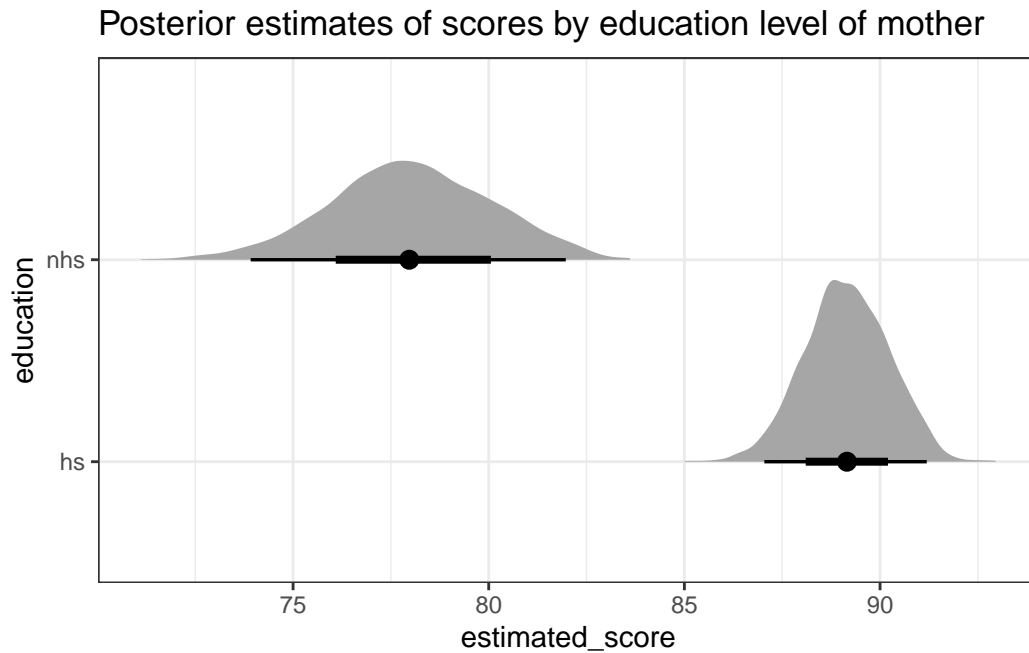
Question 3

- 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?

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.

Question 5

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

Question 6

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