App stat lab exercise 5

```r
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.2     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(rstan)
```

```
Loading required package: StanHeaders

rstan version 2.32.5 (Stan version 2.32.2)

For execution on a local, multicore CPU with excess RAM we recommend calling
options(mc.cores = parallel::detectCores()).
To avoid recompilation of unchanged Stan programs, we recommend calling
rstan_options(auto_write = TRUE)
For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
change `threads_per_chain` option:
rstan_options(threads_per_chain = 1)


Attaching package: 'rstan'
```

```
The following object is masked from 'package:tidyr':

    extract
```

```r
library(tidybayes)
library(here)
```

```
here() starts at /Users/euijinbaek/STA2201
```

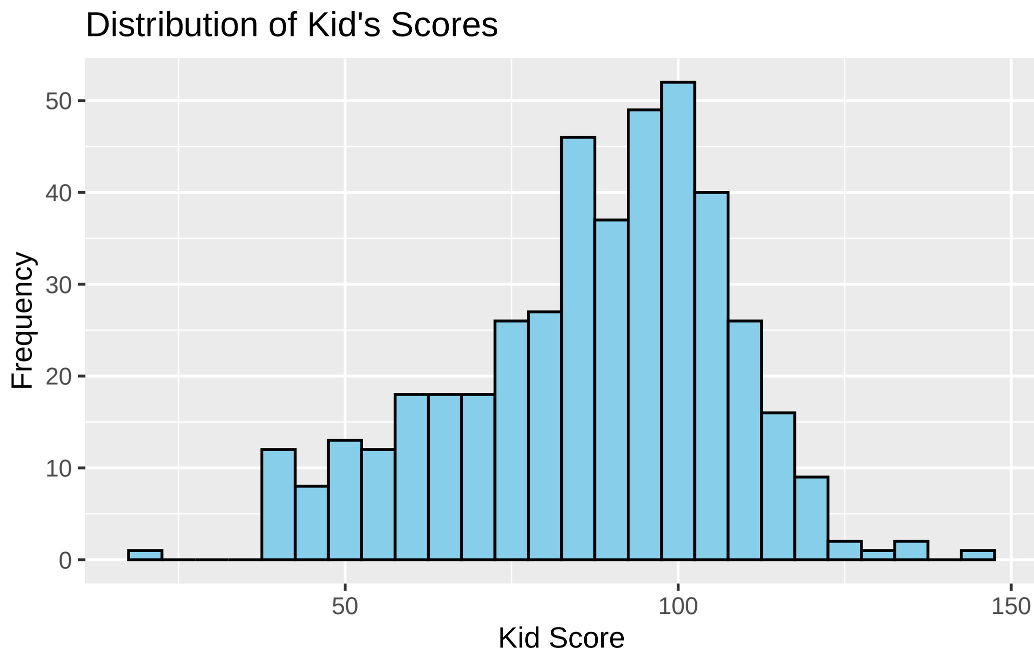
```r
# Data load
kidiq <- read_rds("data/kidiq.RDS")
head(kidiq)
```

```
# A tibble: 6 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
```

1.

We first use historgram to see distribution of Kid's Scores. Since the distribution is not much different from normal distribution, we could assume that Kid's Scores follow normal distribution. (We assume Normal likelihood)

```r
ggplot(kidiq, aes(x = kid_score)) +
  geom_histogram(binwidth = 5, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Kid's Scores", x = "Kid Score", y = "Frequency")
```



Maybe Mother's eduacation level affect kid's score. Let's see Basic statistics. Mean kid score is higer if Mother's education is ar least high school.

```
kidiq |>
  group_by(mom_hs) |>
  summarize(mean_kid_score = mean(kid_score))
```

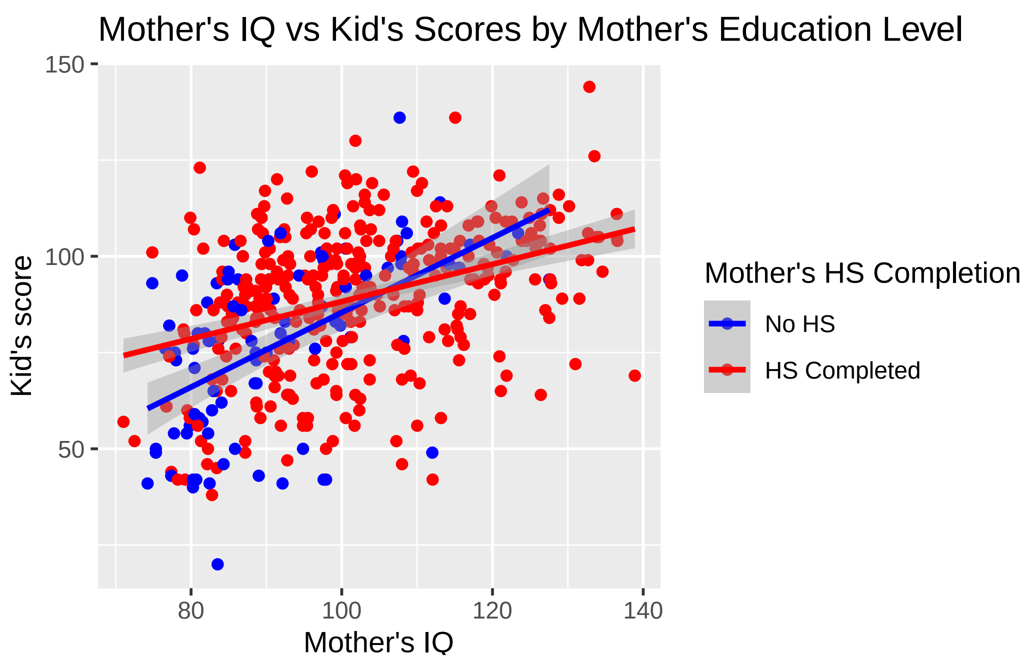
```
# A tibble: 2 x 2
  mom_hs mean_kid_score
   <dbl>          <dbl>
1      0           77.5
2      1           89.3
```

Let's see this relationship in graph. As we can see, there is positive correlation beween kid's score and Mother's IQ overall. The slopes of the two regression lines (one for each group) are positive, which reinforces the observation of a positive correlation. Additionally, the regression line for the mothers who completed high school (red line) is positioned higher than the line for mothers who did not complete high school (blue line), suggesting that completing high school is associated with higher scores for the children, independent of the mother's IQ.

```
kidiq |>
  ggplot(aes(x = mom_iq, y = kid_score, color = as.factor(mom_hs))) +
```

```r
  geom_point() +
  geom_smooth(method = 'lm', aes(group = mom_hs)) +
  labs(title = "Mother's IQ vs Kid's Scores by Mother's Education Level",
       y = "Kid's score",
       x = "Mother's IQ",
       color = "Mother's HS Completion") +
  scale_color_manual(values = c('0' = 'blue', '1' = 'red'),
                     labels = c('0' = 'No HS', '1' = 'HS Completed'))
```

```
`geom_smooth()` using formula = 'y ~ x'
```



Estimating mean, no covariates

```r
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 = "code/models/kids2.stan",
            data = data,
            # reducing the iterations a bit to speed things up
            chains = 3,
            iter = 500)
```

```
Warning in readLines(file, warn = TRUE): incomplete final line found on
'/Users/euijinbaek/STA2201/labs/code/models/kids2.stan'


Trying to compile a simple C file


Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: 'Apple clang version 15.0.0 (clang-1500.0.40.1)'
using SDK: 'MacOSX14.0.sdk'
clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/R
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' 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: 0.003 seconds (Warm-up)
Chain 1:                0.002 seconds (Sampling)
Chain 1:                0.005 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.01 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: 0.003 seconds (Warm-up)
Chain 2:                0.002 seconds (Sampling)
Chain 2:                0.005 seconds (Total)
Chain 2: 

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3: 
Chain 3: Gradient evaluation took 1e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.01 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3: 
Chain 3: 
Chain 3: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 3: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 3: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 3: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 3: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 3: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 3: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 3: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 3: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 3: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 3: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 3: Iteration: 500 / 500 [100%]  (Sampling)
Chain 3: 
Chain 3:  Elapsed Time: 0.005 seconds (Warm-up)
Chain 3:                0.002 seconds (Sampling)
Chain 3:                0.007 seconds (Total)
Chain 3: 
```

```
Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#bulk-ess
```

Look at the summary

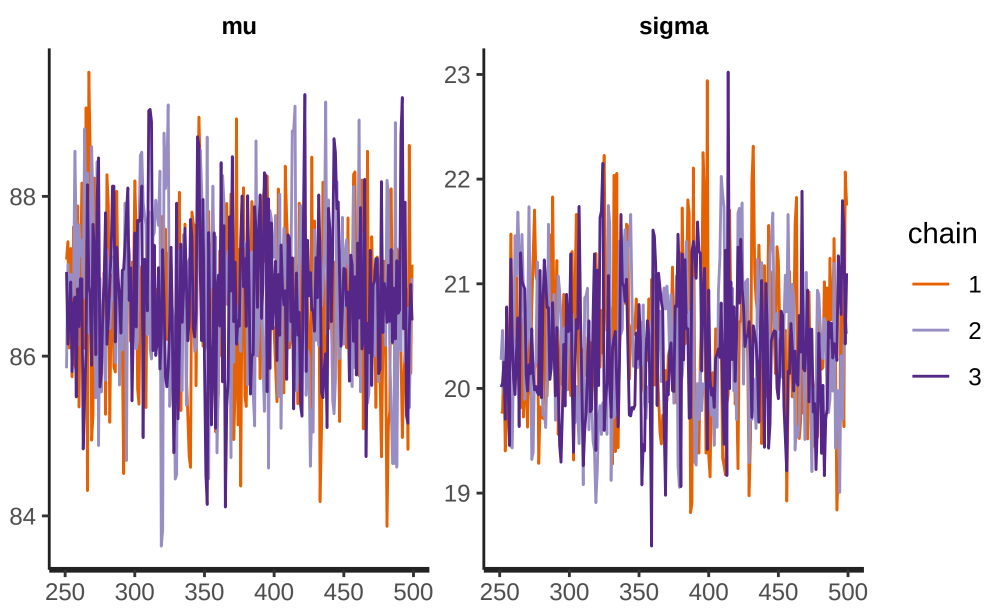
```
fit
```

```
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      86.74    0.04 1.02    84.68    86.08    86.76    87.40    88.76   698
sigma   20.40    0.03 0.68    19.21    19.94    20.37    20.82    21.78   417
lp__ -1525.78    0.06 1.02 -1528.24 -1526.22 -1525.48 -1525.03 -1524.79   276
      Rhat
mu    1.00
sigma 1.00
lp__  1.01

Samples were drawn using NUTS(diag_e) at Fri Feb 16 22:49:58 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

```r
traceplot(fit)
```

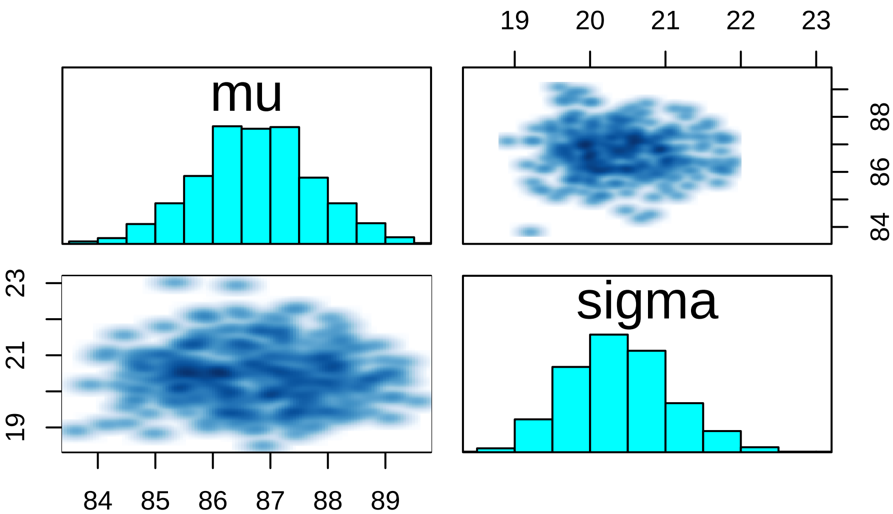


All looks fine.

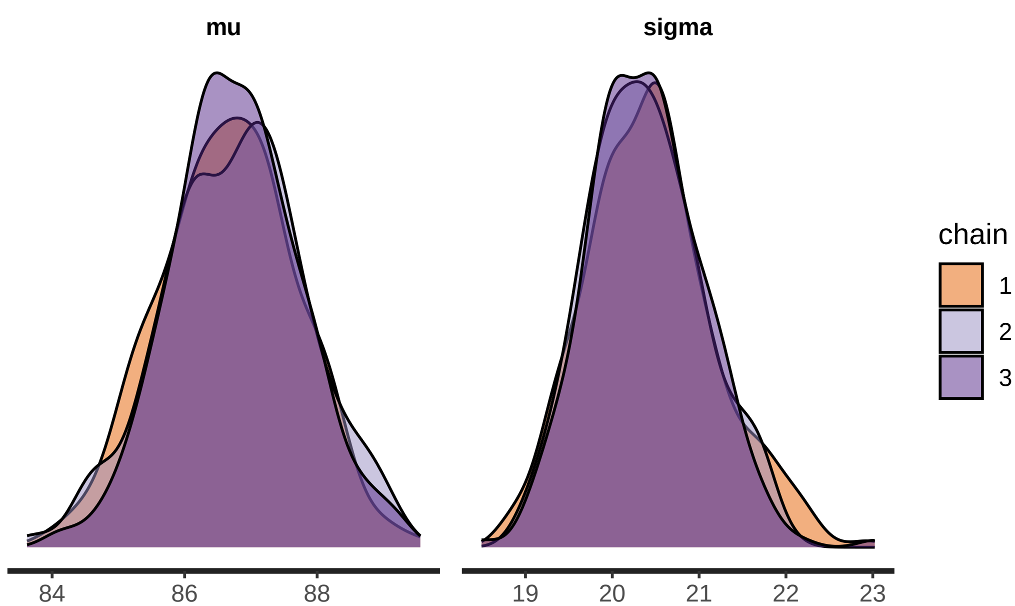
```r
pairs(fit, pars = c("mu", "sigma"))
```

```
Warning in par(usr): argument 1 does not name a graphical parameter
```

```
Warning in par(usr): argument 1 does not name a graphical parameter
```




```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)
names(post_samples)
```

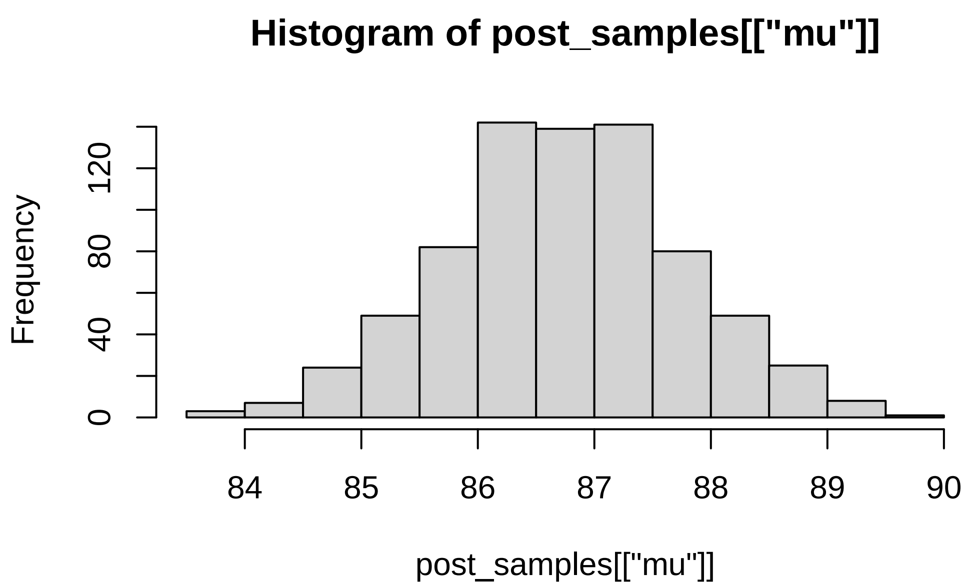
```
[1] "mu"    "sigma" "lp__"
```

```
head(post_samples[["mu"]])
```

```
[1] 85.88881 87.51893 85.75323 86.41366 87.49502 87.04033
```

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] 86.75574
```

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

```
    2.5%
84.68255
```

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

```
  97.5%
88.7553
```

Tidybayes is also very useful:

```r
fit |>
  gather_draws(mu, sigma) |>
  median_qi(.width = 0.8)
```

```
# A tibble: 2 x 7
  .variable .value .lower .upper .width .point .interval
  <chr>      <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
1 mu          86.8   85.4   88.1    0.8 median qi
2 sigma       20.4   19.6   21.3    0.8 median qi
```

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:

```r
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          87.2
 2      1          2     2 mu          87.4
 3      1          3     3 mu          86.1
 4      1          4     4 mu          87.3
 5      1          5     5 mu          85.7
 6      1          6     6 mu          87.6
 7      1          7     7 mu          85.9
 8      1          8     8 mu          85.8
 9      1          9     9 mu          87.9
10      1         10    10 mu          85.4
# i 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  87.2  19.8
 2      1          2     2  87.4  19.8
 3      1          3     3  86.1  20.3
 4      1          4     4  87.3  19.4
 5      1          5     5  85.7  19.8
 6      1          6     6  87.6  19.8
 7      1          7     7  85.9  20.3
 8      1          8     8  85.8  21.5
 9      1          9     9  87.9  20.6
10      1         10    10  85.4  20.4
# i 740 more rows
```

```r
# quickly calculate the quantiles using

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

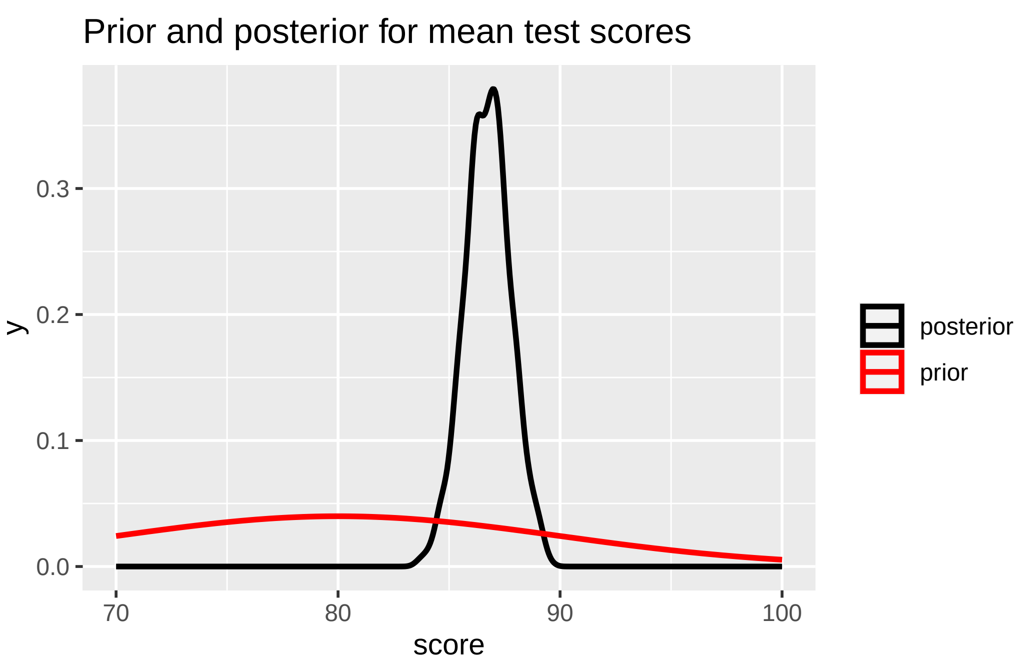
```
# A tibble: 2 x 7
```

```
  .variable .value .lower .upper .width .point .interval
  <chr>      <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
1 mu          86.8   85.4   88.1    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

```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")
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



2.

Let's say we know that relationship are clear and there is little variance. We can encode this by:

```r
sigma0 <- 0.1

data <- list(y = y,
             N = length(y),
             mu0 = mu0,
             sigma0 = sigma0)
fit <- stan(file = "code/models/kids2.stan",
            data = data)
```

```
Warning in readLines(file, warn = TRUE): incomplete final line found on
'/Users/euijinbaek/STA2201/labs/code/models/kids2.stan'
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 4e-06 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.04 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: 0.007 seconds (Warm-up)
Chain 1:                0.007 seconds (Sampling)
Chain 1:                0.014 seconds (Total)
Chain 1:
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.01 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: 0.007 seconds (Warm-up)
Chain 2:                0.007 seconds (Sampling)
Chain 2:                0.014 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 1e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.01 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: 0.007 seconds (Warm-up)
Chain 3:                0.007 seconds (Sampling)
Chain 3:                0.014 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 1e-06 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.01 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.007 seconds (Warm-up)
Chain 4:                0.007 seconds (Sampling)
Chain 4:                0.014 seconds (Total)
Chain 4:
```

Both estimates of mu and sigma are changed.

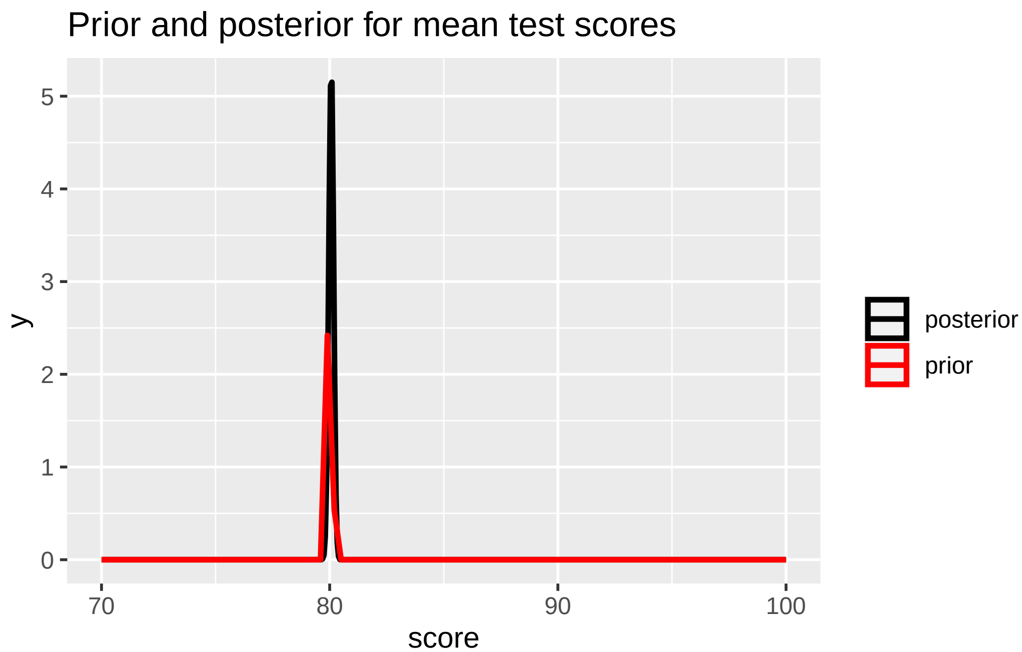
```
fit
```

```
Inference for Stan model: anon_model.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

          mean se_mean   sd     2.5%      25%      50%      75%    97.5% n_eff
```

```
mu        80.07     0.00 0.10     79.87     80.00     80.07     80.13     80.26  3992
sigma     21.40     0.01 0.74     20.01     20.87     21.39     21.90     22.90  4124
lp__   -1548.38     0.03 1.01 -1551.07 -1548.77 -1548.06 -1547.68 -1547.40  1615
       Rhat
mu        1
sigma     1
lp__      1

Samples were drawn using NUTS(diag_e) at Fri Feb 16 22:50:00 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).
```

Compared to the posterior when we set sigma0 $= 10$, we get distribution that has much smaller variance.

```r
# Get the posterior samples for mu and sigma
dsamples <- fit  |>
  gather_draws(mu, sigma)
# Plot
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")
```



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

$$y_i|\mu_i, \sigma^2 \sim N(\mu_i, \sigma^2)$$

$$\mu_i = \alpha + \beta X_i$$

Priors:

$$\alpha \sim N(0, 100^2)$$

$$\beta \sim N(0, 10^2)$$

$$\sigma \sim N(0, 10^2)$$

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.

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

```
Warning in readLines(file, warn = TRUE): incomplete final line found on
'/Users/euijinbaek/STA2201/labs/code/models/kids3.stan'


Trying to compile a simple C file


Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: 'Apple clang version 15.0.0 (clang-1500.0.40.1)'
using SDK: 'MacOSX14.0.sdk'
clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/R
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 4.7e-05 seconds
```

```
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.47 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: 0.063 seconds (Warm-up)
Chain 1:                0.04 seconds (Sampling)
Chain 1:                0.103 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.1 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: 0.06 seconds (Warm-up)
```

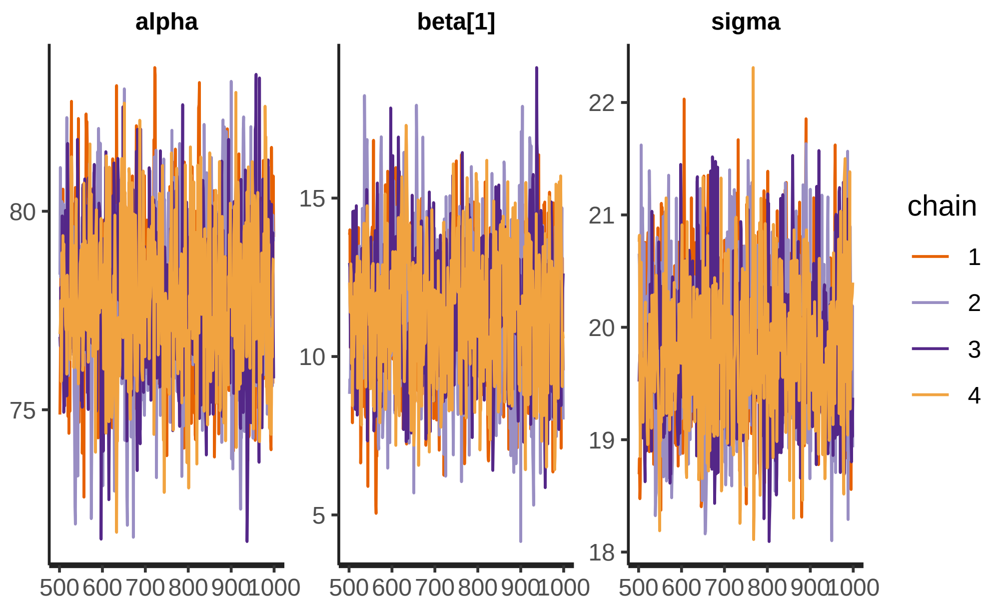
```
Chain 2:                0.043 seconds (Sampling)
Chain 2:                0.103 seconds (Total)
Chain 2: 

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3: 
Chain 3: Gradient evaluation took 1e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.1 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: 0.066 seconds (Warm-up)
Chain 3:                0.041 seconds (Sampling)
Chain 3:                0.107 seconds (Total)
Chain 3: 

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4: 
Chain 4: Gradient evaluation took 9e-06 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.09 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: 0.087 seconds (Warm-up)
Chain 4:                0.044 seconds (Sampling)
Chain 4:                0.131 seconds (Total)
Chain 4:
```

```
fit2
```

```
Inference for Stan model: anon_model.
4 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=2000.

            mean se_mean   sd     2.5%      25%      50%      75%    97.5%
alpha      77.87    0.06 1.95    74.20    76.53    77.90    79.27    81.57
beta[1]    11.38    0.07 2.18     7.27     9.81    11.40    12.91    15.55
sigma      19.83    0.02 0.66    18.64    19.36    19.78    20.25    21.20
lp__    -1514.31    0.04 1.18 -1517.37 -1514.84 -1514.00 -1513.44 -1512.97
        n_eff Rhat
alpha     942 1.00
beta[1]   916 1.00
sigma    1071 1.00
lp__      747 1.01

Samples were drawn using NUTS(diag_e) at Fri Feb 16 22:50:21 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(fit2)
```



3.

Both lm() and fits show the similar coefficient of beta (slope) and alpha (intercept), which are about 11 and 77, respectively.

```
summary(fit2)
```

```
$summary
                mean     se_mean        sd         2.5%          25%          50%
alpha       77.86958 0.06358272 1.9510816    74.200614    76.530174     77.89868
beta[1]     11.37920 0.07211094 2.1820308     7.272061     9.810229     11.39932
sigma       19.82707 0.02009488 0.6575873    18.643141    19.364413     19.77994
lp__     -1514.30725 0.04312568 1.1789003 -1517.370460 -1514.842156 -1514.00018
                 75%       97.5%     n_eff     Rhat
alpha       79.26748    81.56796  941.6135 1.002102
beta[1]     12.90884    15.55160  915.6289 1.001239
sigma       20.25276    21.19934 1070.8677 1.002427
lp__     -1513.43978 -1512.97207  747.2780 1.005588

$c_summary
, , chains = chain:1

         stats
parameter        mean          sd         2.5%          25%          50%
```

```
  alpha      77.95934 1.8880183    74.300963    76.756247    77.89085
  beta[1]    11.25481 2.0886145     7.055529     9.842008    11.34967
  sigma      19.90028 0.6825761    18.719707    19.383784    19.85986
  lp__    -1514.29277 1.1499264 -1517.007526 -1514.905401 -1513.97151
         stats
parameter         75%       97.5%
  alpha      79.20402    81.82523
  beta[1]    12.68686    15.16535
  sigma      20.39787    21.18823
  lp__    -1513.42424 -1512.97635

, , chains = chain:2

         stats
parameter        mean        sd         2.5%          25%          50%
  alpha      77.80700 2.0764272    73.368216    76.424259    77.86924
  beta[1]    11.42456 2.3650606     6.888732     9.800183    11.46754
  sigma      19.81644 0.6826242    18.591756    19.326973    19.76090
  lp__    -1514.41408 1.2845509 -1517.833729 -1514.972239 -1514.05858
         stats
parameter         75%       97.5%
  alpha      79.27680    81.86924
  beta[1]    12.88611    16.40137
  sigma      20.24197    21.25385
  lp__    -1513.49869 -1513.01148

, , chains = chain:3

         stats
parameter        mean        sd        2.5%          25%          50%
  alpha      77.86357 1.9757659    74.29614    76.343550    77.97114
  beta[1]    11.43674 2.1243567     7.58647     9.818212    11.39431
  sigma      19.80122 0.6482061    18.68894    19.349655    19.78242
  lp__    -1514.36375 1.2323343 -1517.87095 -1514.952346 -1514.03212
         stats
parameter         75%       97.5%
  alpha      79.31732    81.50527
  beta[1]    12.94471    15.45840
  sigma      20.19364    21.22624
  lp__    -1513.45101 -1512.95856

, , chains = chain:4
```

```
         stats
parameter        mean        sd         2.5%          25%          50%
  alpha      77.84842 1.8594271    74.253772    76.557583    77.87246
  beta[1]    11.40069 2.1409343     7.696283     9.768627    11.37886
  sigma      19.79033 0.6105092    18.639311    19.403669    19.73649
  lp__    -1514.15839 1.0195506 -1516.920480 -1514.616984 -1513.90723
         stats
parameter         75%       97.5%
  alpha      79.21285    81.32369
  beta[1]    12.99122    15.49366
  sigma      20.21179    20.96367
  lp__    -1513.41292 -1512.93516
```

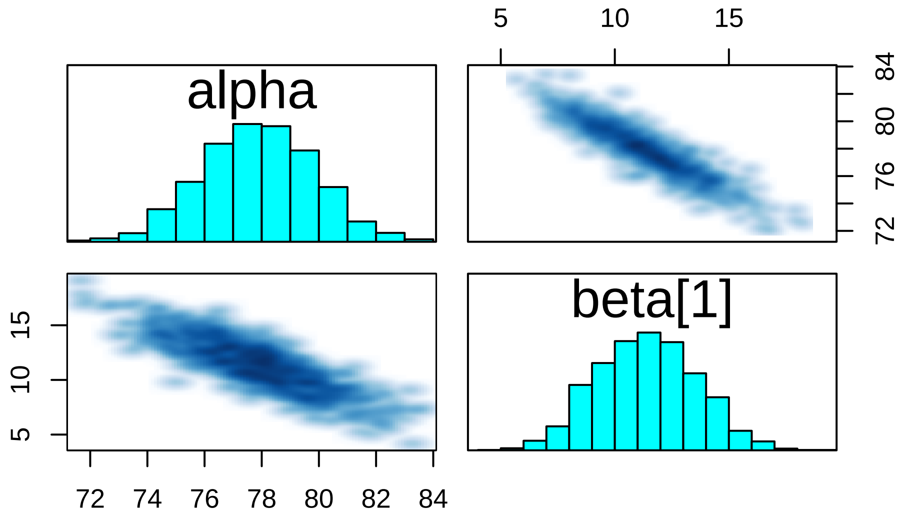
```r
linear <- lm(y~kidiq$mom_hs)
summary(linear)
```

```
Call:
lm(formula = y ~ kidiq$mom_hs)

Residuals:
   Min     1Q Median     3Q    Max
-57.55 -13.32   2.68  14.68  58.45

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    77.548      2.059  37.670  < 2e-16 ***
kidiq$mom_hs   11.771      2.322   5.069 5.96e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 19.85 on 432 degrees of freedom
Multiple R-squared:  0.05613,   Adjusted R-squared:  0.05394
F-statistic: 25.69 on 1 and 432 DF,  p-value: 5.957e-07
```

It seems that they are correlated, which could be problematic. High correlation between parameters can lead to reduced sampling efficiency because we will get narrower results when sampling.

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

```
Warning in par(usr): argument 1 does not name a graphical parameter

Warning in par(usr): argument 1 does not name a graphical parameter
```

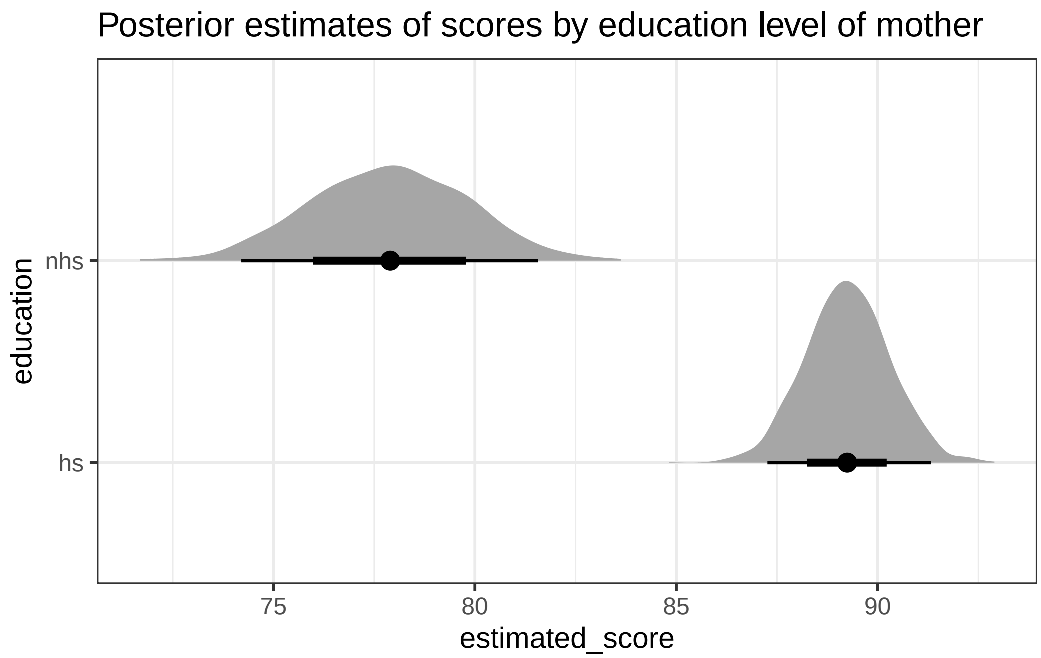


Plotting results

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

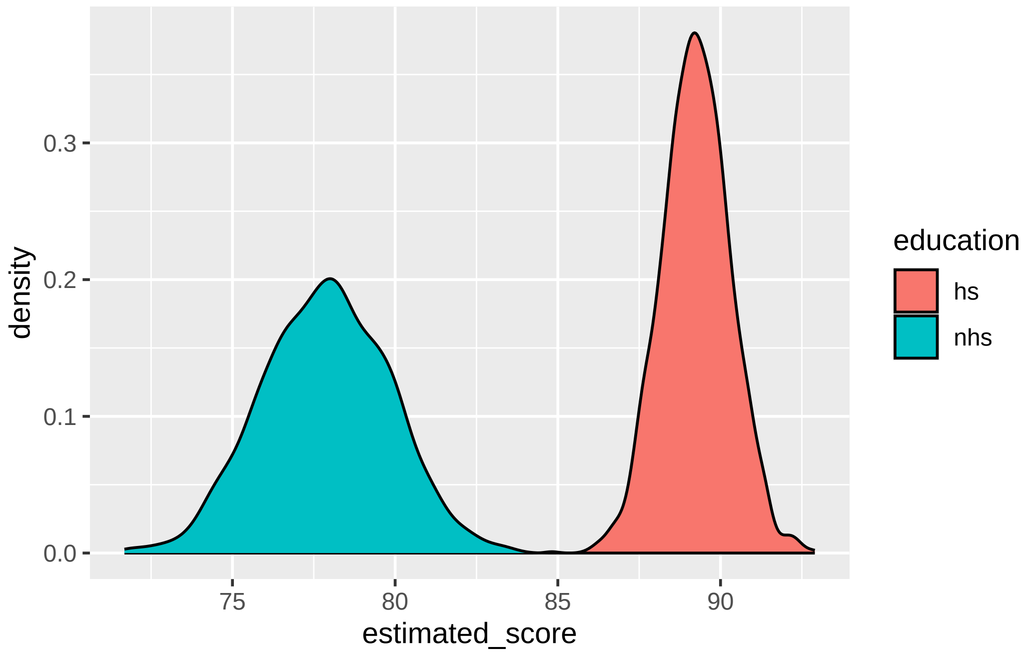
```
fit2 |>
  spread_draws(alpha, beta[k], sigma) |>
     mutate(nhs = alpha, # no high school is just the intercept
          hs = alpha + beta) |>
  select(nhs, hs) |>
  pivot_longer(nhs:hs, names_to = "education", values_to = "estimated_score") |>
  ggplot(aes(y = education, x = estimated_score)) +
  stat_halfeye() +
  theme_bw() +
  ggtitle("Posterior estimates of scores by education level of mother")
```

```
Adding missing grouping variables: `k`
```




```
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(x = estimated_score, fill = education)) +
  geom_density()
```

```
Adding missing grouping variables: `k`
```



4.

Mom's IQ has coefficient of 0.5638947, which suggest that if centered mom's IQ increases by one unit, the expected kid's test score increases about 0.56, holding all other variables constant.

```
# Centering
X <- cbind(kidiq$mom_hs, kidiq$mom_iq - mean(kidiq$mom_iq))
data <- list(y = y,
             N = length(y),
             K = 2,
             X = as.matrix(X))

fit2 <- stan(file = "code/models/kids3.stan",
            data = data,
            iter = 1000)
```

```
Warning in readLines(file, warn = TRUE): incomplete final line found on
'/Users/euijinbaek/STA2201/labs/code/models/kids3.stan'


SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
```

```
Chain 1:
Chain 1: Gradient evaluation took 2.3e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.23 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: 0.094 seconds (Warm-up)
Chain 1:                0.046 seconds (Sampling)
Chain 1:                0.14 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 9e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.09 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: 0.069 seconds (Warm-up)
Chain 2:                0.046 seconds (Sampling)
Chain 2:                0.115 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 9e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:   1 / 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: 0.06 seconds (Warm-up)
Chain 3:                0.045 seconds (Sampling)
Chain 3:                0.105 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 8e-06 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.08 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: 0.08 seconds (Warm-up)
Chain 4:                0.046 seconds (Sampling)
Chain 4:                0.126 seconds (Total)
Chain 4: 
```

```r
summary(fit2)
```

```
$summary
                mean     se_mean         sd          2.5%          25%
alpha     82.2629880 0.056780231 1.82498087    78.8361999    81.035524
beta[1]    5.7689057 0.065306052 2.07579990     1.6527473     4.394039
beta[2]    0.5673964 0.001701716 0.06021509     0.4442557     0.528568
sigma     18.0958025 0.017779593 0.63992428    16.8820276    17.660470
lp__   -1474.4247363 0.055636557 1.45077176 -1478.0405010 -1475.120510
                 50%           75%        97.5%     n_eff      Rhat
alpha     82.2160644    83.4698965    85.953318 1033.0523 1.0019615
beta[1]    5.7806430     7.1285731     9.797362 1010.3320 1.0013195
beta[2]    0.5686704     0.6087015     0.683838 1252.0925 1.0013462
sigma     18.0796648    18.4864798    19.450745 1295.4288 0.9994331
lp__   -1474.0614652 -1473.4041660 -1472.680194  679.9511 1.0043414

$c_summary
, , chains = chain:1

         stats
parameter          mean         sd          2.5%           25%           50%
  alpha      82.3809148 1.92943290    78.8886857    80.9889081    82.3210781
  beta[1]     5.6207502 2.16682892     1.3447890     4.2198757     5.6784750
  beta[2]     0.5722497 0.06320347     0.4462416     0.5294471     0.5718646
  sigma      18.1278201 0.64873244    16.9271882    17.6717071    18.1211719
  lp__    -1474.5470788 1.47376117 -1478.0437686 -1475.2675057 -1474.2797648
         stats
parameter           75%         97.5%
```

```
  alpha      83.6765764    85.9856154
  beta[1]     6.9632377     9.7470421
  beta[2]     0.6119197     0.6986169
  sigma      18.4911626    19.4999415
  lp__    -1473.4219915 -1472.7105520

, , chains = chain:2

         stats
parameter          mean         sd          2.5%           25%           50%
  alpha      82.2257718 1.83373764    78.8262680    80.9447643    82.0811069
  beta[1]     5.8958960 2.04953993     1.9087923     4.5120321     5.9642738
  beta[2]     0.5641275 0.06151765     0.4381999     0.5279423     0.5673596
  sigma      18.1060522 0.68447751    16.8439895    17.6423916    18.0480741
  lp__    -1474.5521175 1.53032867 -1478.3420753 -1475.2373263 -1474.1632638
         stats
parameter           75%         97.5%
  alpha      83.4941751    85.9812707
  beta[1]     7.3711198     9.8053305
  beta[2]     0.6108798     0.6670239
  sigma      18.5219874    19.4690453
  lp__    -1473.4510544 -1472.8069367

, , chains = chain:3

         stats
parameter          mean         sd          2.5%           25%           50%
  alpha      82.2638720 1.75170240    79.2369947    81.0319266    82.1895966
  beta[1]     5.7726053 2.02034291     1.5866264     4.4239216     5.8053891
  beta[2]     0.5636029 0.05413278     0.4600319     0.5275881     0.5636769
  sigma      18.0580415 0.59257567    16.9128588    17.6359656    18.0745197
  lp__    -1474.2296005 1.29251342 -1477.5238475 -1474.9274841 -1473.8918661
         stats
parameter           75%        97.5%
  alpha      83.4052958    85.966372
  beta[1]     7.1421030     9.620892
  beta[2]     0.5998191     0.665958
  sigma      18.4428474    19.187422
  lp__    -1473.3158101 -1472.634043

, , chains = chain:4

         stats
```

```
parameter          mean          sd          2.5%          25%          50%
  alpha      82.1813934 1.77946158    78.6240498    81.1569495    82.200554
  beta[1]     5.7863714 2.06050169     1.8677266     4.4474010     5.760751
  beta[2]     0.5696055 0.06133637     0.4407034     0.5288136     0.571073
  sigma      18.0912961 0.63038368    16.8883767    17.6742629    18.048181
  lp__    -1474.3701484 1.47470116 -1478.2197566 -1474.9230911 -1473.988818
         stats
parameter           75%         97.5%
  alpha      83.3604047    85.6894062
  beta[1]     6.9569270    10.1429686
  beta[2]     0.6140754     0.6853418
  sigma      18.4801283    19.3449771
  lp__    -1473.3735486 -1472.6736586
```

5.

The result agrees with 'lm()'

```r
linear <- lm(y ~ X[,1] +  X[,2])
summary(linear)
```

```
Call:
lm(formula = y ~ X[, 1] + X[, 2])

Residuals:
    Min      1Q  Median      3Q     Max 
-52.873 -12.663   2.404  11.356  49.545 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 82.12214    1.94370  42.250  < 2e-16 ***
X[, 1]       5.95012    2.21181   2.690  0.00742 ** 
X[, 2]       0.56391    0.06057   9.309  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 18.14 on 431 degrees of freedom
Multiple R-squared:  0.2141,    Adjusted R-squared:  0.2105 
F-statistic: 58.72 on 2 and 431 DF,  p-value: < 2.2e-16
```

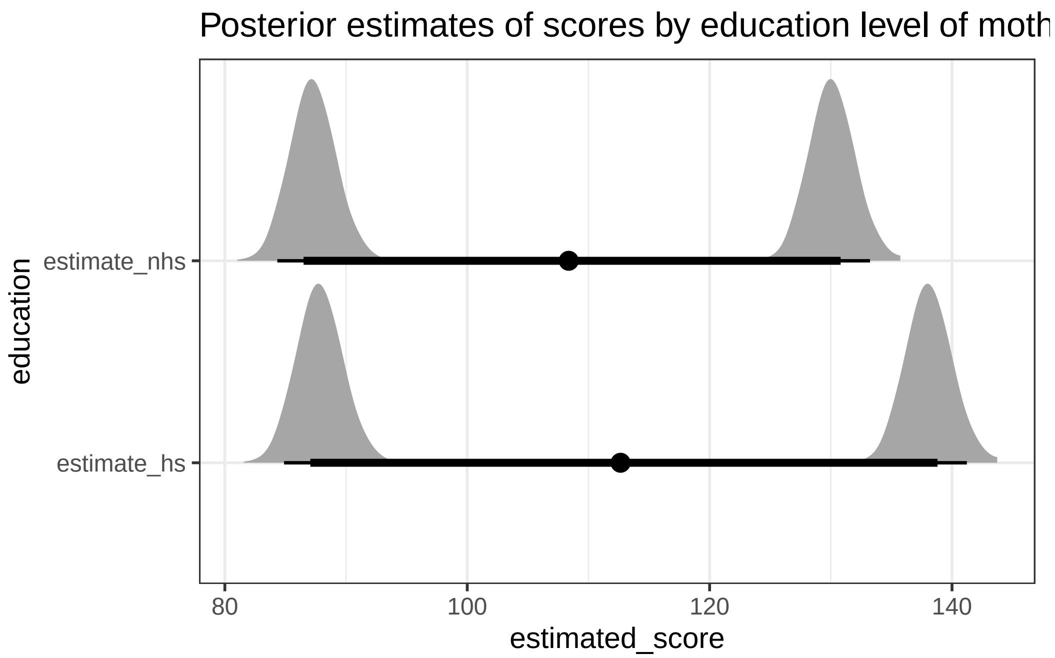
6.

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

```
# Using `spread_draws` to extract the relevant draws from the model.
posterior <- fit2 |>
  spread_draws(alpha, beta[k], sigma) |>
  # Adjust for mother's IQ of 110 (centered around the mean)
  mutate(iq_adjustment = 110 - mean(kidiq$mom_iq)) |>
  mutate(estimate_nhs = alpha + iq_adjustment * beta[2], # Estimate for mothers without hi
         estimate_hs = alpha + iq_adjustment * beta[2] + beta[1]) |>  # Estimate for mothe
  select(estimate_nhs, estimate_hs) |>
  pivot_longer(estimate_nhs:estimate_hs, names_to = "education", values_to = "estimated_sc
```

```
Adding missing grouping variables: `k`
```

```
# Plot the estimates
ggplot(posterior, aes(y = education, x = estimated_score)) +
  stat_halfeye() +
  theme_bw() +
  ggtitle("Posterior estimates of scores by education level of mother for IQ = 110")
```



7.

```
samples <- extract(fit2)
pred <- samples[["alpha"]] + samples[["beta"]][,1] + (95-mean(kidiq$mom_iq))*samples[["bet
sigma <- samples[["sigma"]]
y_pred <- tibble(y_pred = rnorm(length(sigma), mean = pred, sd = sigma))
ggplot(y_pred, aes(y_pred)) + geom_histogram(fill = "skyblue", col = "blue") + ggtitle("Di
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```