Week 5: Bayesian linear regression and introduction to Stan

12/03/24

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("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

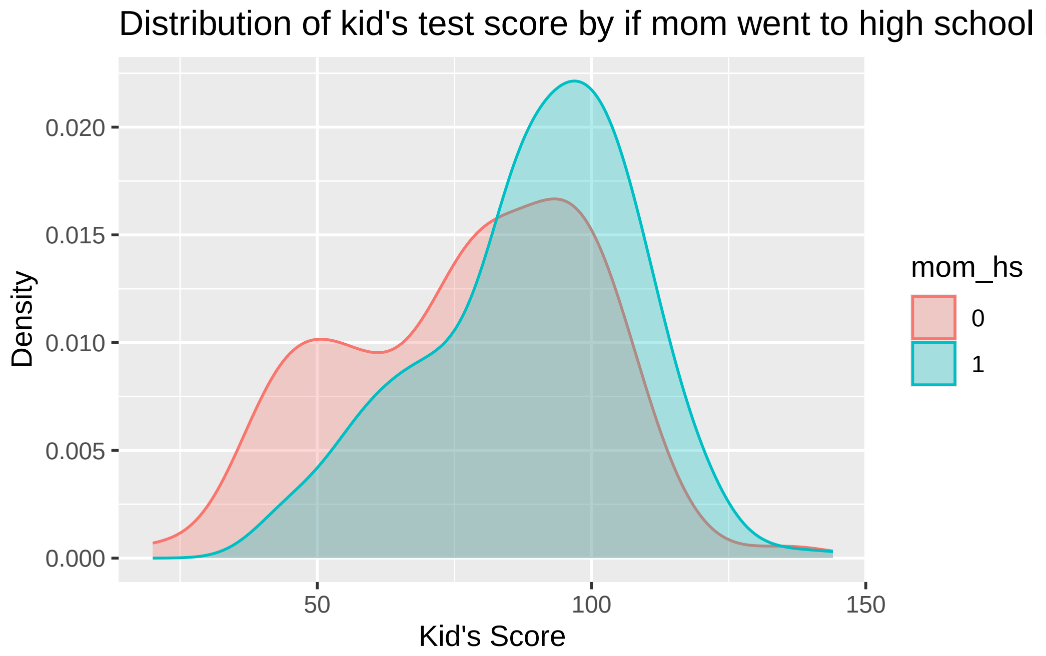
Question 1

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

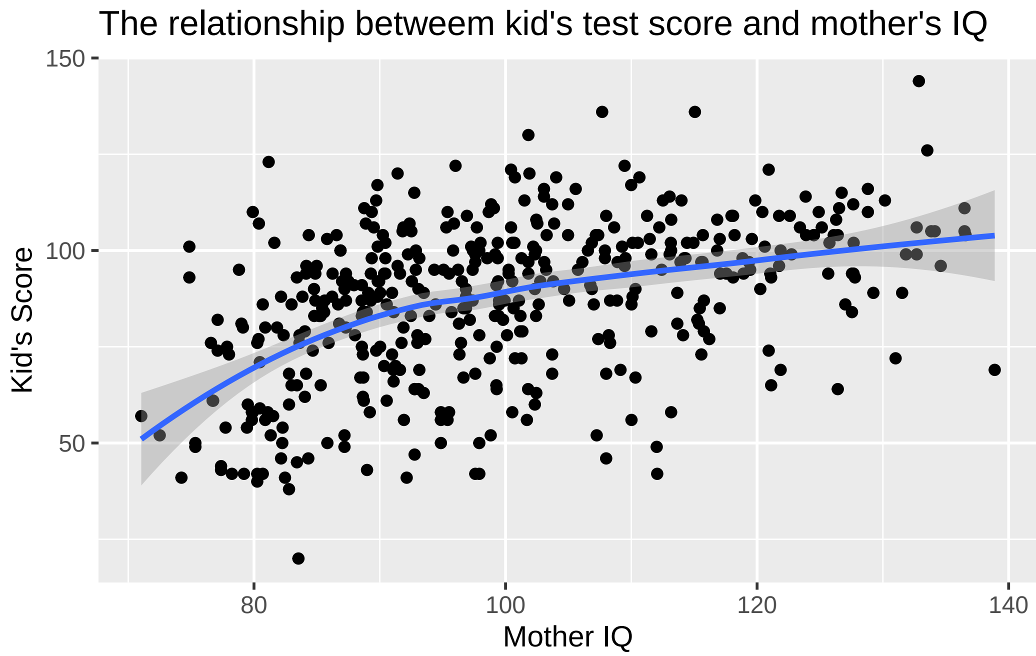
I first find it would be interesting to see how does the distribution of kid's test score impacted by mom's high school status. According to the plot below, For Mom who went to high school, the kid_score distribution on average shifts a bit higher and the density for those with higher test scores seems to be higher as well. For example, when test score is 100, the density for kid's whose mom went to high scholl has a higher test scores than those who didn't.

```
kidiq$mom_hs <- as.character(kidiq$mom_hs)
ggplot(kidiq, aes(x=kid_score, color = mom_hs, fill=mom_hs)) +
  geom_density(alpha=0.3) +
  labs(x = "Kid's Score", y="Density", title = "Distribution of kid's test score by if mom
```



Then it would be interesting to explore the relationship between mother's IQ and kid's score. From the scatter plot below we see that there seems to have a positive relationship between the two. An increase in Mother's IQ will likely result in an increase in kid's test score.

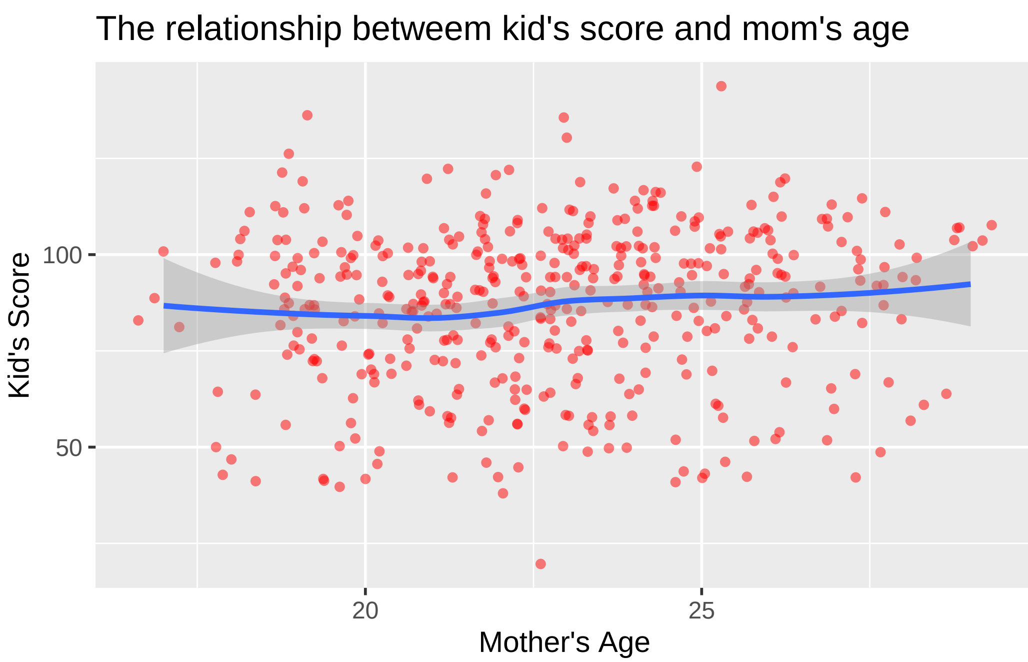
```
ggplot(kidiq, aes(x=mom_iq, y=kid_score)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Mother IQ", y="Kid's Score", title = "The relationship betweem kid's test scor
```



3 - The relationship between the kid's test score and the mother's age

Next I explored whether there are some relationships between the kid's test score and the mother's age. This curve is rather flat, indicating that there is no clear relationship that these two variables are related in anyay.

```
ggplot(kidiq, aes(x=mom_age, y=kid_score)) +
  geom_point(position = "jitter", alpha=0.5, shape = 16, color="red") +
  geom_smooth() +
  labs(x = "Mother's Age", y="Kid's Score", title = "The relationship betweem kid's score
```



Estimating mean, no covariates

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

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

priors:

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

$$\mu \sim N(\mu_0, \sigma_0^2)$$

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

# named list to input for stan function
data <- list(y = y,
```

```r
               N = length(y),
               mu0 = mu0,
               sigma0 = sigma0)
```

Now we can run the model:

```r
fit <- stan(file = here("kids2.stan"),
            data = data,
            # reducing the iterations a bit to speed things up
            chains = 3,
            iter = 500)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: ‘Apple clang version 15.0.0 (clang-1500.3.9.4)’
using SDK: ‘MacOSX14.4.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
#include <cmath>
         ^~~~~~~
1 error generated.
make: *** [foo.o] Error 1

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 / 500 [  0%]  (Warmup)
Chain 1: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 1: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 1: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 1: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 1: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 1: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 1: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 1: Iteration: 350 / 500 [ 70%]  (Sampling)
```

```
Chain 1: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 1: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 1: Iteration: 500 / 500 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.004 seconds (Warm-up)
Chain 1:                0.001 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:                0.001 seconds (Sampling)
Chain 2:                0.004 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: 0.006 seconds (Warm-up)
Chain 3:                0.001 seconds (Sampling)
Chain 3:                0.007 seconds (Total)
Chain 3:
```

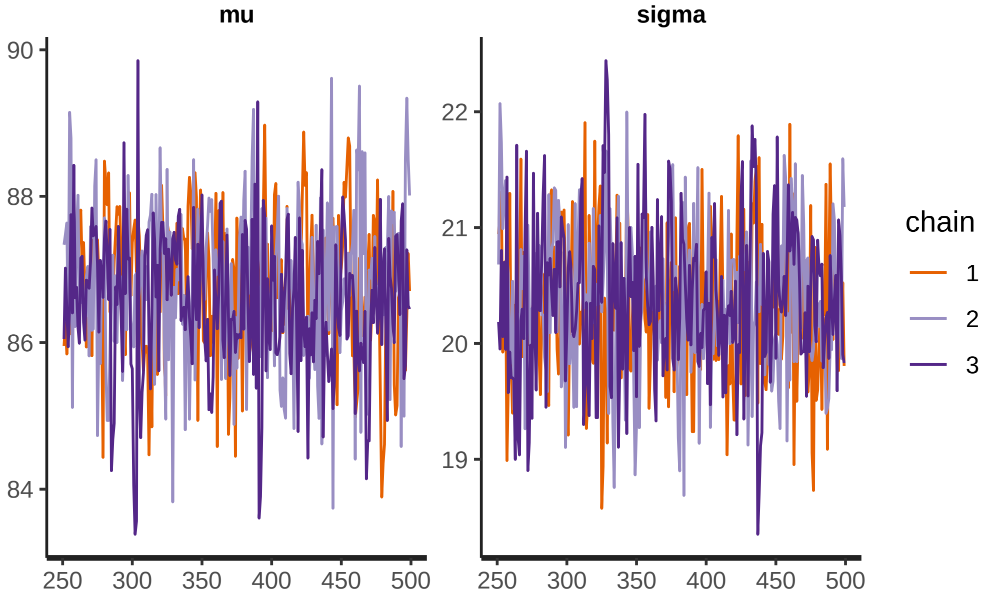
Look at the summary

```r
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       86.70    0.05 0.99    84.62    86.10    86.74    87.41    88.50   363
sigma    20.36    0.03 0.65    19.11    19.93    20.32    20.80    21.63   551
lp__  -1525.73    0.06 1.00 -1528.60 -1526.08 -1525.41 -1525.03 -1524.78   300
      Rhat
mu    1.01
sigma 1.00
lp__  1.00

Samples were drawn using NUTS(diag_e) at Tue Mar 12 23:00:29 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).
```

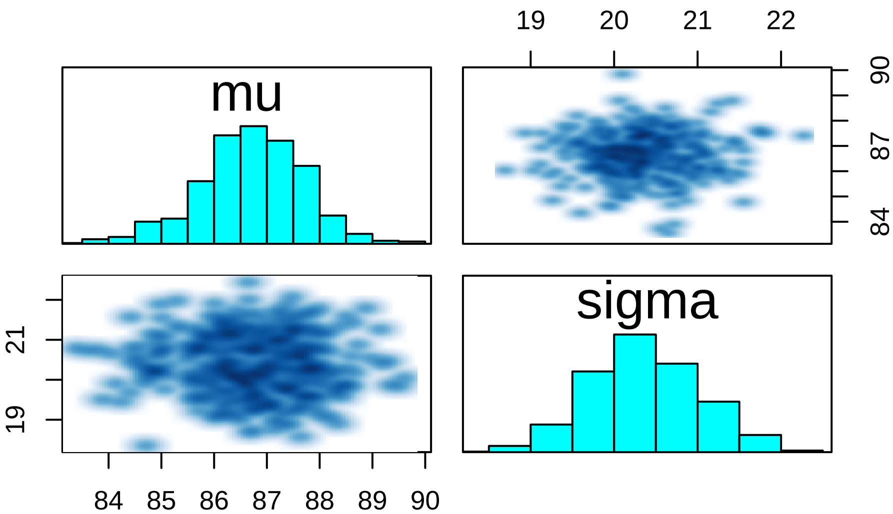
Traceplot

```r
traceplot(fit)
```

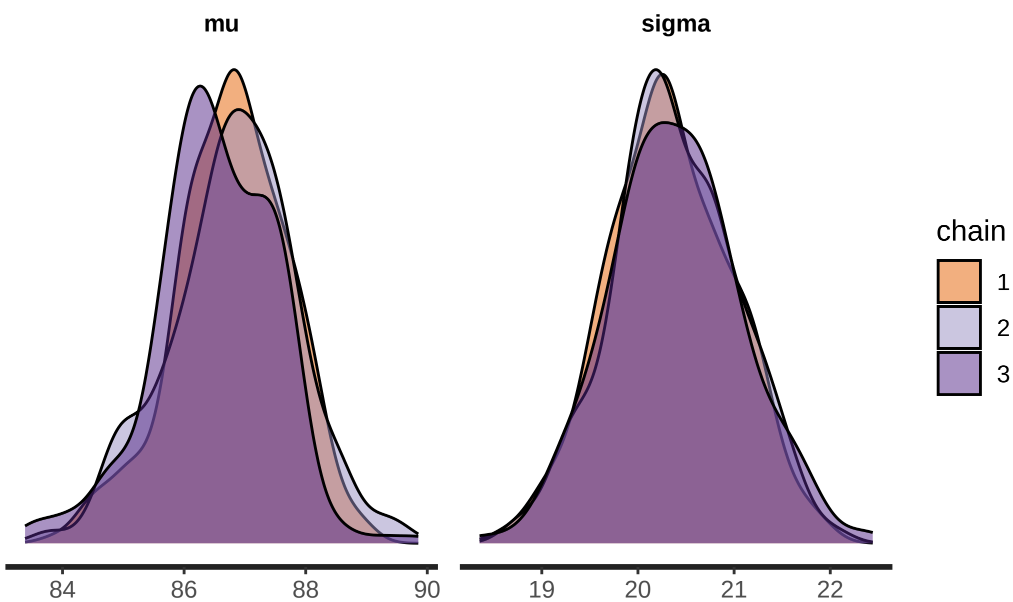


All looks fine.

```
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)
names(post_samples)
```

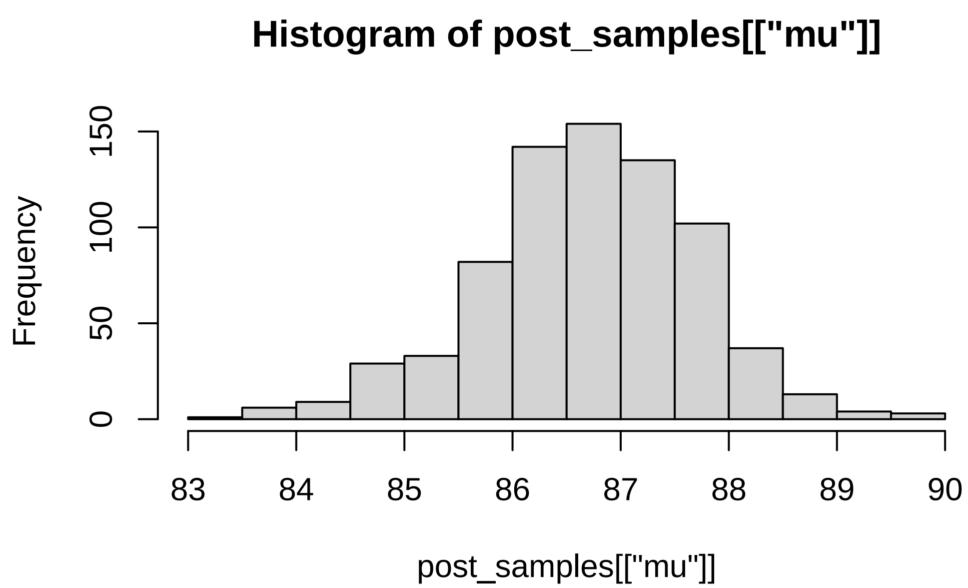
```
[1] "mu"    "sigma" "lp__"
```

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

```
[1] 87.01659 87.05295 87.13596 85.82149 86.89247 86.58275
```

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] 86.74203
```

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

```
    2.5%
84.61519
```

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

```
   97.5%
88.50291
```

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          86.7   85.5   87.9    0.8 median qi
2 sigma       20.3   19.5   21.2    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          86.0
 2      1          2     2 mu          86.3
 3      1          3     3 mu          85.8
 4      1          4     4 mu          86.2
 5      1          5     5 mu          87.7
 6      1          6     6 mu          86.1
 7      1          7     7 mu          86.5
 8      1          8     8 mu          87.0
 9      1          9     9 mu          87.1
10      1         10    10 mu          87.1
# i 1,490 more rows
```

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

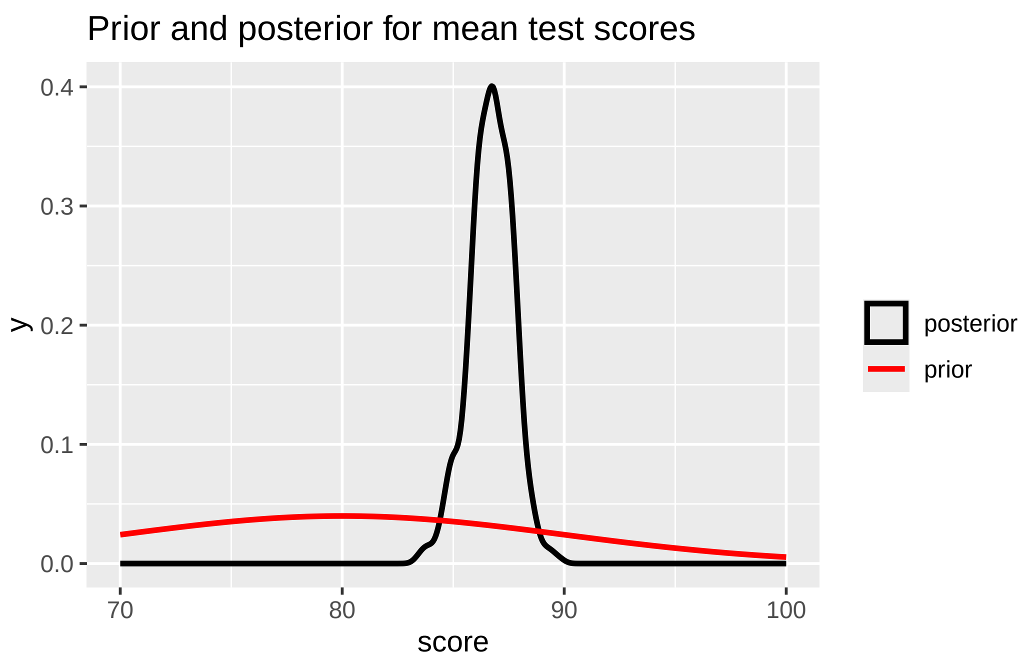
```
# A tibble: 750 x 5
   .chain .iteration .draw    mu sigma
    <int>      <int> <int> <dbl> <dbl>
 1      1          1     1  86.0  20.9
 2      1          2     2  86.3  21.2
 3      1          3     3  85.8  21.3
 4      1          4     4  86.2  19.9
 5      1          5     5  87.7  20.7
 6      1          6     6  86.1  20.7
 7      1          7     7  86.5  19.0
 8      1          8     8  87.0  19.7
 9      1          9     9  87.1  21.3
10      1         10    10  87.1  19.8
# 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          86.7   85.5   87.9    0.8 median qi
2 sigma       20.3   19.5   21.2    0.8 median qi
```

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

```r
dsamples |>
  filter(.variable == "mu") |>
  ggplot(aes(.value, color = "posterior")) + geom_density(size = 1) +
  xlim(c(70, 100)) +
  stat_function(fun = dnorm,
        args = list(mean = mu0,
                    sd = sigma0),
        aes(colour = 'prior'), size = 1) +
  scale_color_manual(name = "", values = c("prior" = "red", "posterior" = "black")) +
  ggtitle("Prior and posterior for mean test scores") +
  xlab("score")
```



Question 2

Change the prior to be much more informative (by changing the standard deviation to be 0.1). Rerun the model. Do the estimates change? Plot the prior and posterior densities.

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

# sd 0.1
sigma1 <- 0.1

data1 <- list(y = y,
              N = length(y),
              mu0 = mu0,
              sigma0 = sigma1)

fit1 <- stan(file = here("kids2.stan"),
             data = data1,
             chains = 3,
             iter = 2000)
```

```
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:                0.006 seconds (Sampling)
Chain 1:                0.013 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)
```

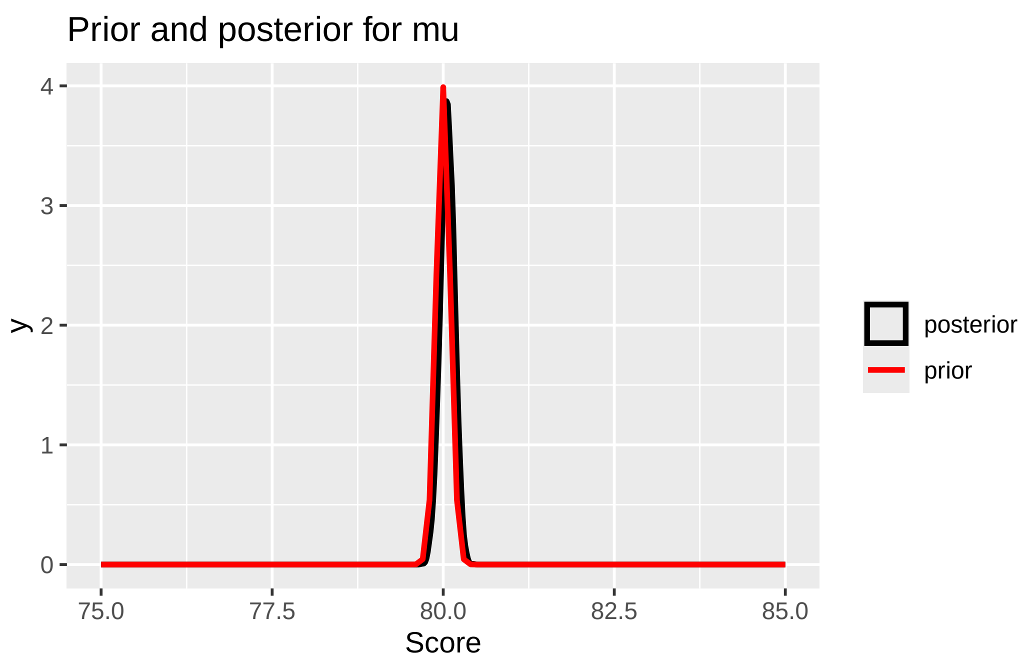
```
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2: 
Chain 2:  Elapsed Time: 0.007 seconds (Warm-up)
Chain 2:                0.006 seconds (Sampling)
Chain 2:                0.013 seconds (Total)
Chain 2: 

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:                0.006 seconds (Sampling)
Chain 3:                0.013 seconds (Total)
Chain 3: 
```

```r
summary(fit1)$summary
```

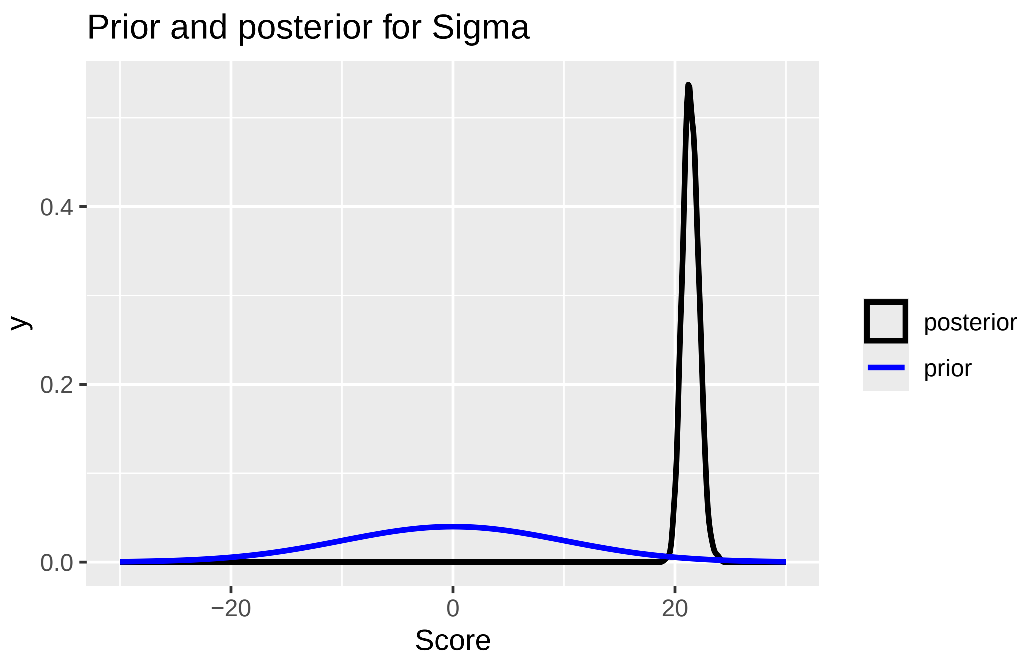
```
             mean     se_mean        sd        2.5%         25%         50%
mu       80.06273 0.001920354 0.1025624    79.85565    79.99489    80.06252
sigma    21.42672 0.013480755 0.7299952    20.06770    20.92765    21.39004
lp__  -1548.40993 0.027053977 1.0400636 -1551.16715 -1548.80291 -1548.10127
              75%       97.5%    n_eff      Rhat
mu       80.13408    80.25777 2852.419 0.9999005
sigma    21.91106    22.87379 2932.321 0.9998908
lp__  -1547.66723 -1547.39561 1477.943 1.0020831
```

We see that the estimation does not change much compared to the first one.

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

sigma1 <- 0.1

dsamples1 <- fit1   %>%
  gather_draws(mu, sigma) # gather = long format

dsamples1  %>%
  filter(.variable == "mu")  %>%
  ggplot(aes(.value, color = "posterior")) + geom_density(size = 1) +
  xlim(c(75, 85)) +
  stat_function(fun = dnorm,
        args = list(mean = mu0,
                    sd = sigma1),
        aes(colour = 'prior'), size = 1) +
  scale_color_manual(name = "", values = c("prior" = "red", "posterior" = "black")) +
  ggtitle("Prior and posterior for mu") +
  xlab("Score")
```



```
dsamples1  %>%
  filter(.variable == "sigma")  %>%
  ggplot(aes(.value, color = "posterior")) + geom_density(size = 1) +
  xlim(c(-30, 30)) +
  stat_function(fun = dnorm,
        args = list(mean = 0,
                    sd = 10),
        aes(colour = 'prior'), size = 1) +
  scale_color_manual(name = "", values = c("prior" = "blue", "posterior" = "black")) +
  ggtitle("Prior and posterior for Sigma") +
  xlab("Score")
```



Here we see the big difference! We see that with this more informative prior, it is evident that the posterior distribution for mu becomes more closer to the piror, indicating a better fit. The posterior shape for sigma is changed as well although the prior is on mu only. This tells us that getting informative priors is really important when fitting a bayesian model.

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.

```
kidiq <- read_rds("kidiq.RDS")
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("kids3.stan"),
            data = data,
            iter = 5000)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: ‘Apple clang version 15.0.0 (clang-1500.3.9.4)’
using SDK: ‘MacOSX14.4.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
#include <cmath>
         ^~~~~~~
1 error generated.
make: *** [foo.o] Error 1

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

```
Chain 1:
Chain 1: Gradient evaluation took 4.2e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.42 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 1: Iteration: 2500 / 5000 [ 50%]  (Warmup)
Chain 1: Iteration: 2501 / 5000 [ 50%]  (Sampling)
Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.183 seconds (Warm-up)
Chain 1:                0.185 seconds (Sampling)
Chain 1:                0.368 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 / 5000 [  0%]  (Warmup)
Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 2: Iteration: 2500 / 5000 [ 50%]  (Warmup)
Chain 2: Iteration: 2501 / 5000 [ 50%]  (Sampling)
Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
```

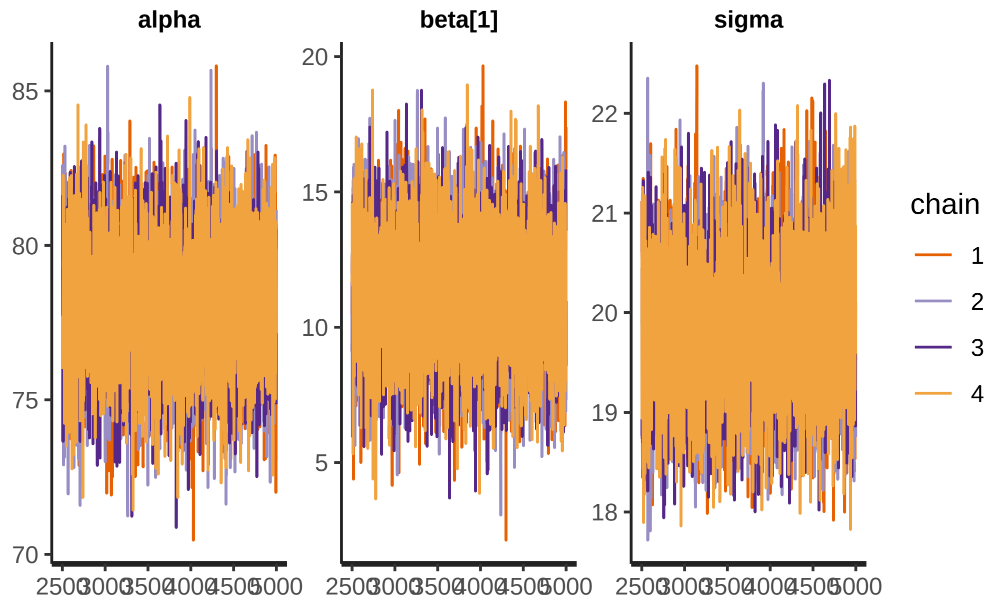
```
Chain 2:
Chain 2:  Elapsed Time: 0.174 seconds (Warm-up)
Chain 2:                0.174 seconds (Sampling)
Chain 2:                0.348 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 1.2e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.12 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 3: Iteration: 2500 / 5000 [ 50%]  (Warmup)
Chain 3: Iteration: 2501 / 5000 [ 50%]  (Sampling)
Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.193 seconds (Warm-up)
Chain 3:                0.188 seconds (Sampling)
Chain 3:                0.381 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 7e-06 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
```

```
Chain 4: Iteration: 2500 / 5000 [ 50%]  (Warmup)
Chain 4: Iteration: 2501 / 5000 [ 50%]  (Sampling)
Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 4: 
Chain 4:  Elapsed Time: 0.194 seconds (Warm-up)
Chain 4:                0.177 seconds (Sampling)
Chain 4:                0.371 seconds (Total)
Chain 4: 
```

```
fit2
```

```
Inference for Stan model: anon_model.
4 chains, each with iter=5000; warmup=2500; thin=1; 
post-warmup draws per chain=2500, total post-warmup draws=10000.

            mean se_mean   sd     2.5%      25%      50%      75%    97.5%
alpha      77.98    0.03 2.00    74.13    76.62    78.00    79.35    81.88
beta[1]    11.19    0.03 2.25     6.84     9.65    11.17    12.71    15.55
sigma      19.82    0.01 0.67    18.55    19.36    19.80    20.26    21.18
lp__    -1514.36    0.02 1.21 -1517.46 -1514.91 -1514.05 -1513.47 -1512.97
        n_eff Rhat
alpha    4111    1
beta[1]  4180    1
sigma    5081    1
lp__     3680    1

Samples were drawn using NUTS(diag_e) at Tue Mar 12 23:00:51 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)
```



Question 3

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

```r
summary(fit)$summary
```

```
             mean     se_mean        sd        2.5%         25%         50%
mu       86.70197 0.05202157 0.9907555    84.61519    86.10170    86.74203
sigma    20.36474 0.02779428 0.6525984    19.11364    19.92767    20.31918
lp__  -1525.72673 0.05772506 0.9991705 -1528.60219 -1526.08031 -1525.41406
              75%       97.5%    n_eff      Rhat
mu       87.40846    88.50291 362.7155 1.0100520
sigma    20.79995    21.63386 551.2915 0.9991825
lp__  -1525.02947 -1524.77976 299.6059 1.0014587
```

```r
# lm
model2 <- lm(kidiq$kid_score ~ kidiq$mom_hs)
summary(model2)
```

```
Call:
lm(formula = kidiq$kid_score ~ kidiq$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
```

```r
summary(fit2)$summary[1:2,1]
```

```
   alpha  beta[1] 
77.98448 11.18918 
```

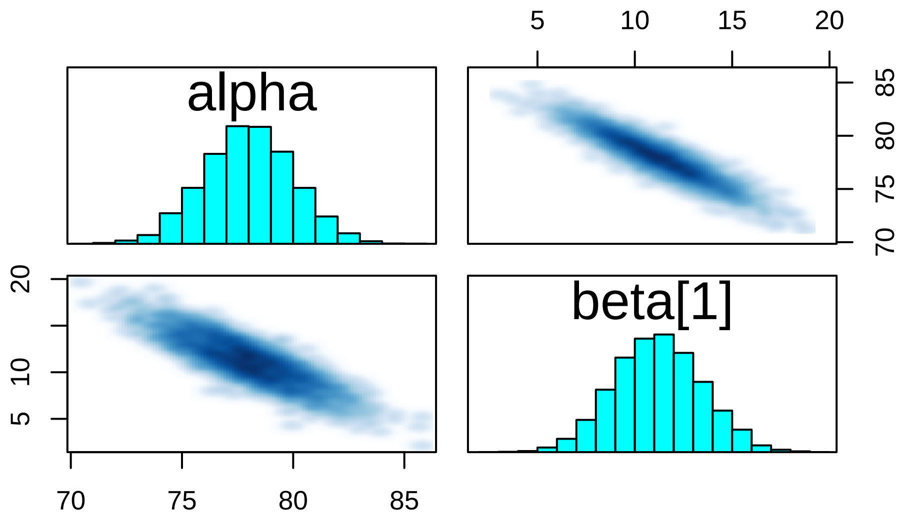
```r
summary(model2)$coefficients[,"Estimate"]
```

```
 (Intercept) kidiq$mom_hs 
    77.54839     11.77126 
```

The two methods yields similar coeeficients estimates as shown above. The Bayesian posterior mean for mu is 86.68008 and the Bayesian posterior mean for sigma is 20.37315. The residual standard error (an estimate of the standard deviation of the error term, sigma in Bayesian terms) from lm is 19.85, similar to bayesian.

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



Well the above plot shows that there is a negative correlation relationship between the slope and the intercept. As the intercepts gets larger, the slop becomes smaller. The correlation between the two should be close to -1. This means that there would be a slight problem as a small change in slope can change the intercept and thus making it harder to sample.

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

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.

```
kidiq$mom_iq_meanadj <- kidiq$mom_iq - mean(kidiq$mom_iq)

X <- cbind(kidiq$mom_hs, kidiq$mom_iq_meanadj)
```

```r
y <- kidiq$kid_score  # Assuming kid_score is the response variable
data3 <- list(y = y, N = length(y), X = X, K = ncol(X))
```

```r
fit3 <- stan(file = here("kids3.stan"),
             data = data3,
             iter = 5000)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 1.1e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 1: Iteration: 2500 / 5000 [ 50%]  (Warmup)
Chain 1: Iteration: 2501 / 5000 [ 50%]  (Sampling)
Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.226 seconds (Warm-up)
Chain 1:                0.229 seconds (Sampling)
Chain 1:                0.455 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 / 5000 [  0%]  (Warmup)
```

```
Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 2: Iteration: 2500 / 5000 [ 50%]  (Warmup)
Chain 2: Iteration: 2501 / 5000 [ 50%]  (Sampling)
Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.202 seconds (Warm-up)
Chain 2:                0.223 seconds (Sampling)
Chain 2:                0.425 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 / 5000 [  0%]  (Warmup)
Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 3: Iteration: 2500 / 5000 [ 50%]  (Warmup)
Chain 3: Iteration: 2501 / 5000 [ 50%]  (Sampling)
Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.206 seconds (Warm-up)
Chain 3:                0.214 seconds (Sampling)
Chain 3:                0.42 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 / 5000 [  0%]  (Warmup)
Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
Chain 4: Iteration: 2500 / 5000 [ 50%]  (Warmup)
Chain 4: Iteration: 2501 / 5000 [ 50%]  (Sampling)
Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.219 seconds (Warm-up)
Chain 4:                0.203 seconds (Sampling)
Chain 4:                0.422 seconds (Total)
Chain 4:
```

Here is a summary of the model:

```
summary(fit3)$summary
```

```
                mean    se_mean         sd          2.5%           25%
alpha     82.3034724 0.026874072 1.91279358    78.5525907    81.0295799
beta[1]    5.7043306 0.030437921 2.16357091     1.4375105     4.2488608
beta[2]    0.5648792 0.000748904 0.06022564     0.4476376     0.5244779
sigma     18.1226138 0.007399806 0.62368166    16.9513867    17.6925219
lp__    -1474.4512761 0.023277437 1.45082334 -1478.1207734 -1475.1684847
                 50%           75%         97.5%    n_eff      Rhat
alpha     82.3104425    83.6023964    86.0283539 5066.047 1.0009729
beta[1]    5.7130236     7.1271976    10.0339288 5052.569 1.0003088
beta[2]    0.5644586     0.6060581     0.6823646 6467.116 0.9998084
sigma     18.1073552    18.5318869    19.3988586 7103.711 0.9996563
lp__    -1474.1322462 -1473.3776100 -1472.6662036 3884.711 1.0016796
```

This model implicate the following observations: 1. We see that the coefficient of the mean_centered IQ is approximately 0.565, which means that given all other variables unchanged, the kid's test score will likely increase by 0.565 points if mom's IQ rise by 1 unit. 2. The intercept shows that the base test score that the kid will have when no high school mom and a mean IQ.

Question 5

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

```
summary(fit3)$summary
```

```
                mean     se_mean         sd        2.5%         25%
alpha     82.3034724 0.026874072 1.91279358   78.5525907   81.0295799
beta[1]    5.7043306 0.030437921 2.16357091    1.4375105    4.2488608
beta[2]    0.5648792 0.000748904 0.06022564    0.4476376    0.5244779
sigma     18.1226138 0.007399806 0.62368166   16.9513867   17.6925219
lp__    -1474.4512761 0.023277437 1.45082334 -1478.1207734 -1475.1684847
                 50%          75%        97.5%    n_eff      Rhat
alpha     82.3104425   83.6023964   86.0283539 5066.047 1.0009729
beta[1]    5.7130236    7.1271976   10.0339288 5052.569 1.0003088
beta[2]    0.5644586    0.6060581    0.6823646 6467.116 0.9998084
sigma     18.1073552   18.5318869   19.3988586 7103.711 0.9996563
lp__    -1474.1322462 -1473.3776100 -1472.6662036 3884.711 1.0016796
```

```
model3 <- lm(kidiq$kid_score ~ kidiq$mom_hs + kidiq$mom_iq_meanadj)
summary(model3)
```

```
Call:
lm(formula = kidiq$kid_score ~ kidiq$mom_hs + kidiq$mom_iq_meanadj)

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 ***
kidiq$mom_hs          5.95012    2.21181   2.690  0.00742 **
```

```
kidiq$mom_iq_meanadj  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
```

```r
summary(fit3)$summary[1:3,1]
```

```
     alpha    beta[1]    beta[2]
82.3034724  5.7043306  0.5648792
```

```r
summary(model3)$coefficients[,"Estimate"]
```

```
         (Intercept)         kidiq$mom_hs kidiq$mom_iq_meanadj
           82.122143             5.950117             0.563906
```

Well we see that the coefficient from the Bayesian model and the MLE estimator are pretty similar, so is the case with the standardized error. The stan model result is confirmed with the lm() approach.

Question 6

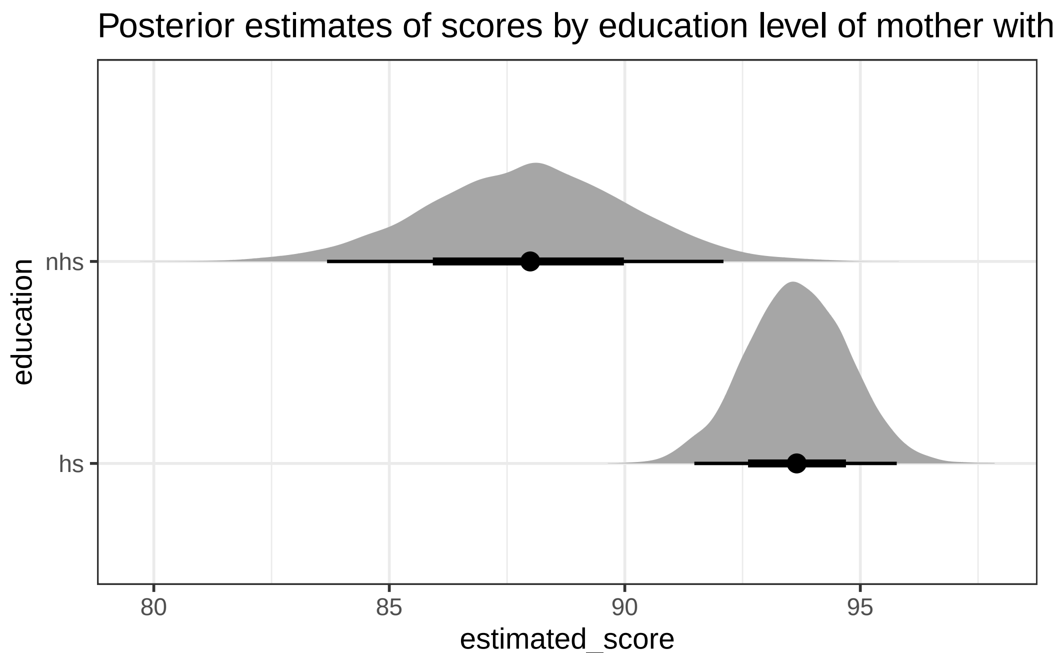
Plot the posterior estimates of scores by education of mother for mothers who have an IQ of 110.

```r
110-mean(kidiq$mom_iq)
```

```
[1] 10
```

```r
fit3 %>%
  spread_draws(alpha, beta[k], sigma) %>%
  pivot_wider(names_from = k, names_prefix = "beta", values_from = beta) %>%  # Transforms
  mutate(
    nhs = alpha + beta2 * 10,  # Calculates the estimated score for non-high school gradua
    hs = alpha + beta1 + beta2 * 10  # Calculates the estimated score for high school grad
```

```
  ) %>%
  select(nhs, hs) %>%
  pivot_longer(nhs:hs, names_to = "education", values_to = "estimated_score") %>%
  ggplot(aes(y = education, x = estimated_score)) +
  stat_halfeye() +
  theme_bw() +
  ggtitle("Posterior estimates of scores by education level of mother with IQ 110")
```



Question 7

Generate and plot (as a histogram) samples from the posterior predictive distribution for a new kid with a mother who graduated high school and has an IQ of 95.

```
x_diff <- 95- mean(kidiq$mom_iq)

post_samples3 <- extract(fit3)
sigma <- post_samples3$sigma

pred <- post_samples3$alpha + post_samples3$beta[,1] + post_samples3$beta[,2] *x_diff

new_sample <- rnorm(length(sigma), mean = pred, sd = sigma)
```

```r
hist(new_sample, main="Histogram of kid's test score")
```