Week 5: Bayesian linear regression and introduction to Stan

12/02/23

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
# ... with 424 more rows
```

As well as the kid's test scores, we have a binary variable indicating whether or not the mother completed high school, the mother's IQ and age.

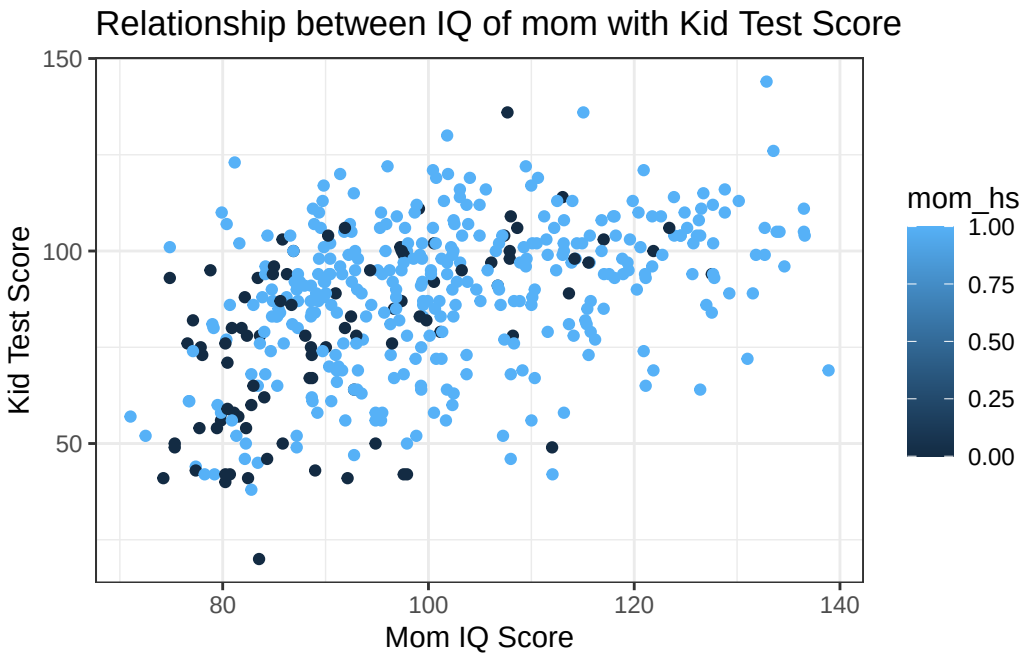
Descriptives

Question 1

Use plots or tables to show three interesting observations about the data. Remember:

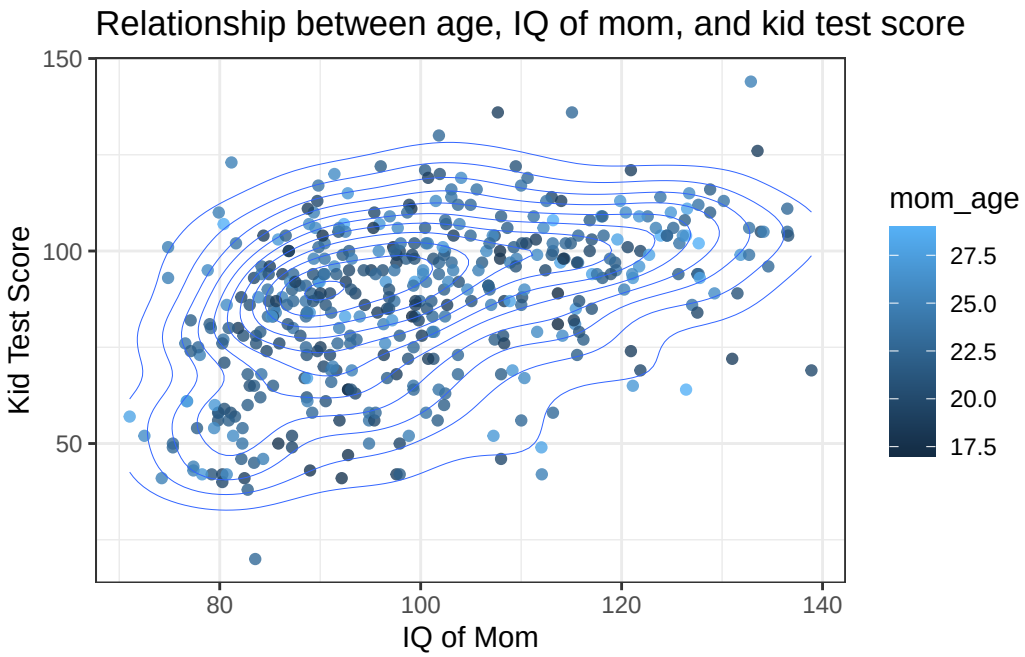
- Explain what your graph/ tables show
- Choose a graph type that's appropriate to the data type

```
kidiq %>%
  ggplot(aes(x = mom_iq, y = kid_score, color = mom_hs)) +
  geom_point() +
  theme_bw() +
  xlab('Mom IQ Score') +
  ylab('Kid Test Score') +
  ggtitle('Relationship between IQ of mom with Kid Test Score')
```



This shows us the relationship between IQ score and the kid's test score, just like we saw in class. However, now we have an added layer of the mom's secondary school education. We see that most people have graduated high school, but of the parents who have not, many tend to have lower IQs. Thus, it seems reasonable that IQ is correlated with the ability to graduate high school. We will check for this correlation later, when we do a correlation plot. Also we notice that there seems to be a (slight) positive linear trend between the mother's IQ and the Kid's test score. So we have some reason to believe that there is a positive correlation. This could possibly influence the choice of our prior (although we'd be cheating since we are implicitly using information from the data).

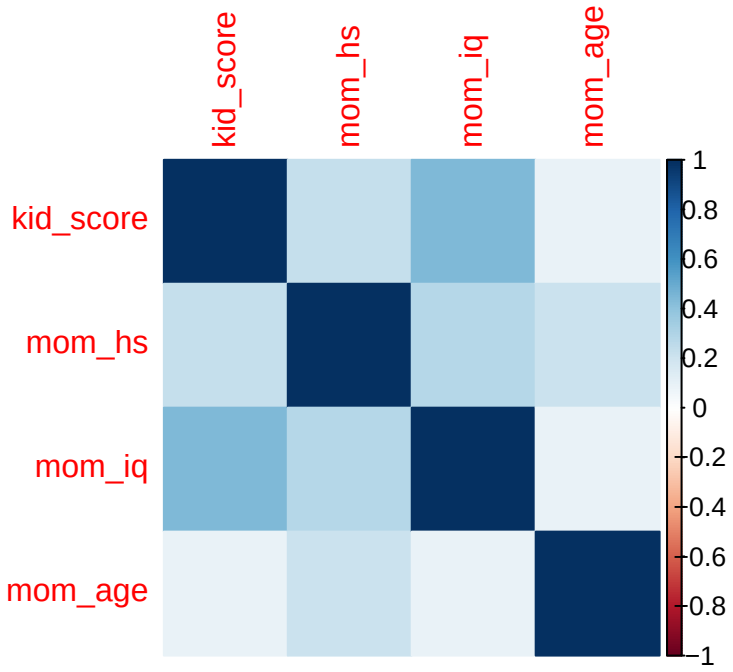
```
kidiq %>%
  ggplot(aes(x = mom_iq, y = kid_score, color = mom_age)) +
  geom_point(alpha = 0.8) +
  stat_density_2d(linewidth = 0.1) +
  theme_bw() +
  xlab('IQ of Mom') +
  ylab('Kid Test Score') +
  ggtitle('Relationship between age, IQ of mom, and kid test score')
```



This time we overlay with the age of the mother (instead of high school status). Note that age is actually treated as a discrete variable, so plotting directly on a scatterplot is not advised. Instead it represents the colour. We see that there is not much variability in the age of mothers. The range of values is only from 17.5 to about 28. This doesn't account for mothers in their thirties, for instance. This should be kept in mind when we run our analysis. In addition, there is not a clear trend between age and the test score, as it seems that all ages seem to be present at all levels of IQ and the kid's test score. Let's look at this closer with a correlation plot.

Finally, to avoid issues with multi-collinearity, let's look at pairwise correlations.

```
library(corrplot)
corrplot(cor(kidiq), method = 'color')
```



We see no two values are too correlated, except `mom_iq` and `kid_score` seems to be somewhat correlated. However it is not so much as to be concerned. In addition, the age of the mom does not seem to be correlated much to the kids test score, as we observed in the previous plot. With this in mind, we don't worry too much about co-linear covariates, so we continue.

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)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 2.2e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.22 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: 0.023 seconds (Warm-up)
Chain 1:                0.006 seconds (Sampling)
Chain 1:                0.029 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 6e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
```

```
Chain 2:
Chain 2: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 2: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 2: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 2: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 2: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 2: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 2: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 2: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 2: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 2: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 2: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 2: Iteration: 500 / 500 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.014 seconds (Warm-up)
Chain 2:                0.005 seconds (Sampling)
Chain 2:                0.019 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 5e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 3: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 3: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 3: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 3: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 3: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 3: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 3: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 3: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 3: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 3: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 3: Iteration: 500 / 500 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.016 seconds (Warm-up)
Chain 3:                0.006 seconds (Sampling)
Chain 3:                0.022 seconds (Total)
Chain 3:
```

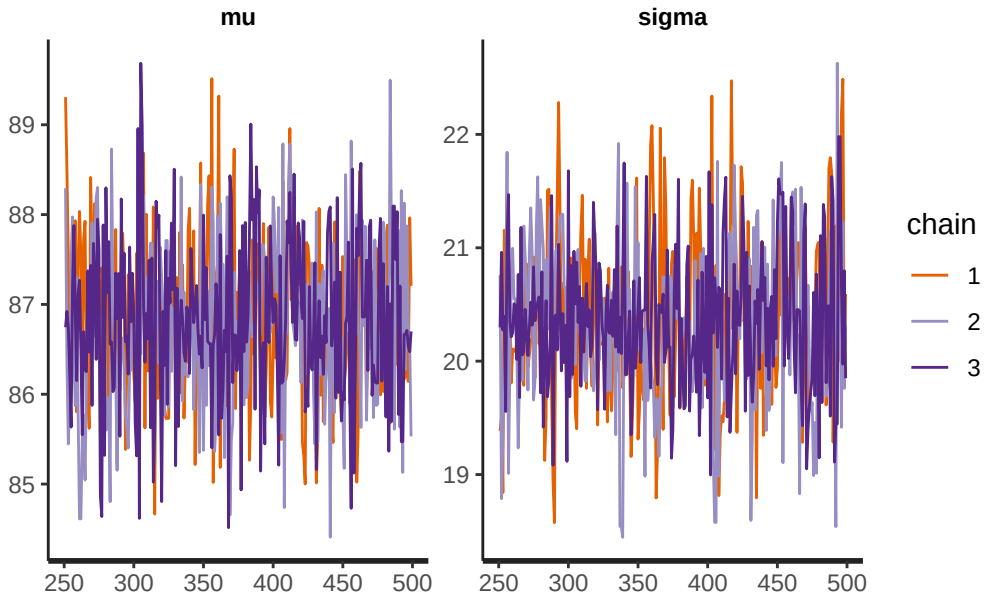
Look at the summary

```
fit
```

```
Inference for Stan model: anon_model.
3 chains, each with iter=500; warmup=250; thin=1;
post-warmup draws per chain=250, total post-warmup draws=750.

          mean se_mean   sd     2.5%      25%      50%      75%    97.5% n_eff
mu       86.81    0.03 0.91    85.04    86.22    86.80    87.45    88.51   784
sigma    20.38    0.03 0.68    19.01    19.91    20.36    20.81    21.72   580
lp__  -1525.69    0.05 0.94 -1528.47 -1526.04 -1525.40 -1525.02 -1524.77   302
      Rhat
mu       1
sigma    1
lp__     1

Samples were drawn using NUTS(diag_e) at Sun Feb 12 12:57:05 2023.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

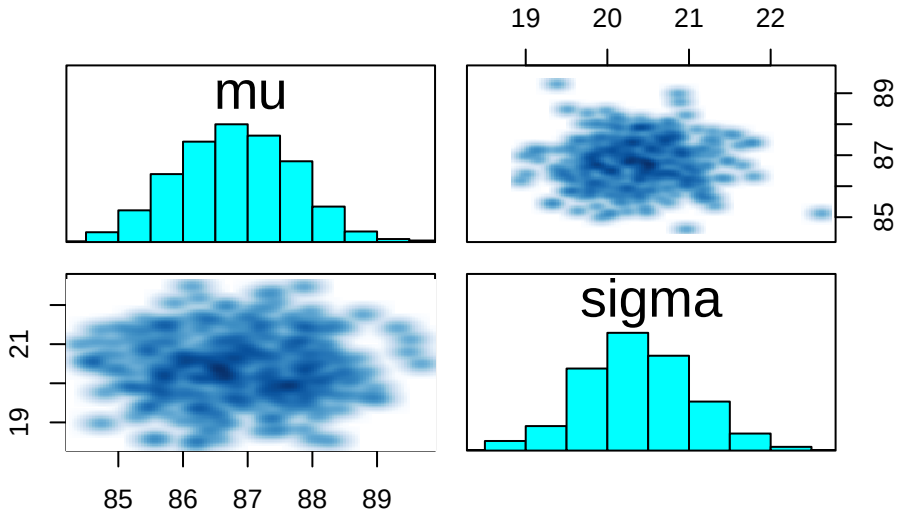
Traceplot

```
traceplot(fit)
```

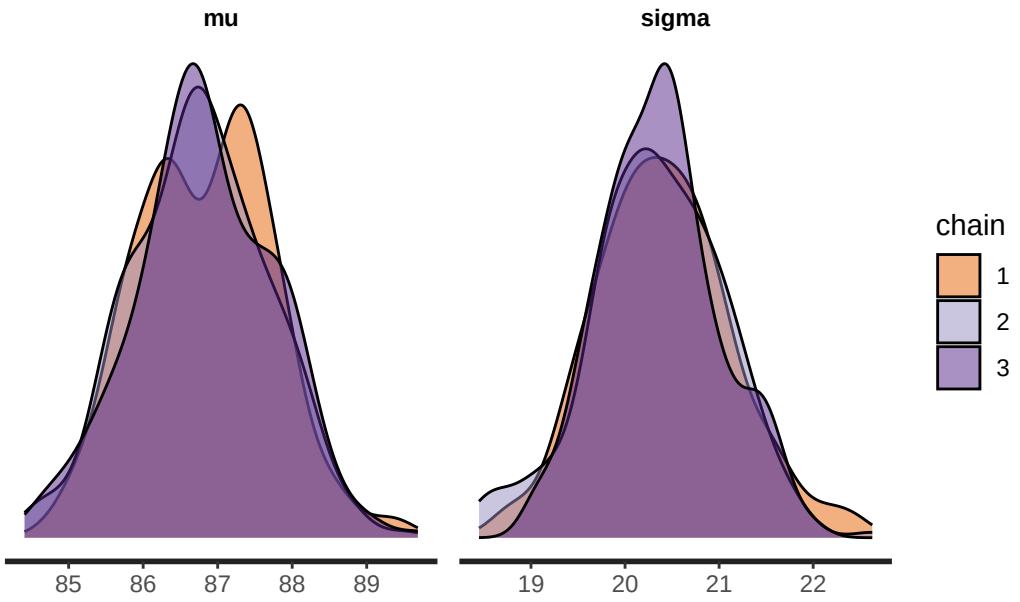


All looks fine.

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




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



Understanding output

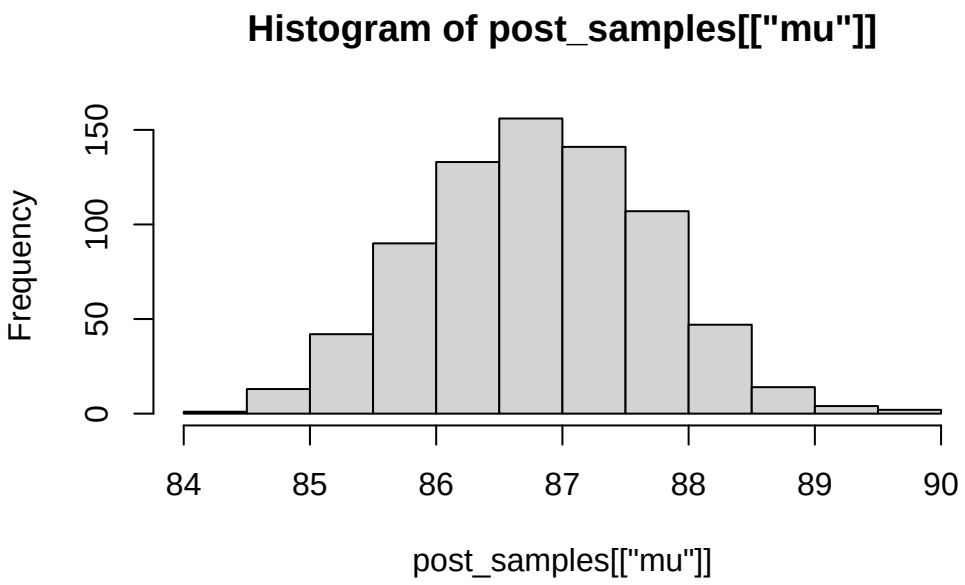
What does the model actually give us? A number of samples from the posteriors. To see this, we can use `extract` to get the samples.

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

```
[1] 85.46387 86.74449 87.66414 86.79540 85.66442 87.22876
```

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] 86.79799
```

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

```
   2.5%
85.0386
```

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

```
   97.5%
88.50622
```

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 tibble
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          89.3
 2      1          2     2 mu          88.5
 3      1          3     3 mu          87.5
 4      1          4     4 mu          85.7
 5      1          5     5 mu          86.0
 6      1          6     6 mu          86.5
 7      1          7     7 mu          86.7
 8      1          8     8 mu          87.9
 9      1          9     9 mu          85.8
10      1         10    10 mu          87.4
# ... with 1,490 more rows
```

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

```
# A tibble: 750 x 5
   .chain .iteration .draw    mu sigma
    <int>      <int> <int> <dbl> <dbl>
 1      1          1     1  89.3  19.4
 2      1          2     2  88.5  19.5
 3      1          3     3  87.5  18.8
 4      1          4     4  85.7  21.2
 5      1          5     5  86.0  20.9
 6      1          6     6  86.5  21.1
 7      1          7     7  86.7  20.3
 8      1          8     8  87.9  20.6
 9      1          9     9  85.8  19.8
10      1         10    10  87.4  20.1
# ... with 740 more rows
```

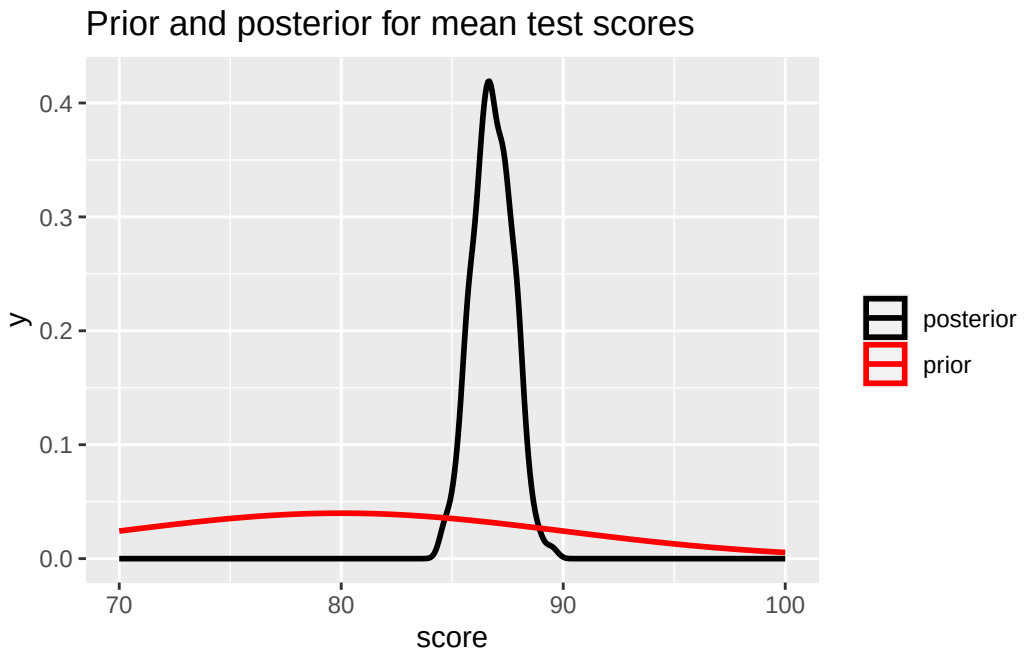
```
  # quickly calculate the quantiles using

  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          86.8   85.6   88.0    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.

```
mu0 <- 80
sigma0 <- 0.1 # New

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

Now we can re-run the model:

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

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 5e-06 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.05 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: 0.006 seconds (Warm-up)
Chain 1:                0.004 seconds (Sampling)
Chain 1:                0.01 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' 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: 0.008 seconds (Warm-up)
Chain 2:                0.004 seconds (Sampling)
Chain 2:                0.012 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 2e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.02 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:                0.004 seconds (Sampling)
Chain 3:                0.009 seconds (Total)
Chain 3:
```

Look at the summary

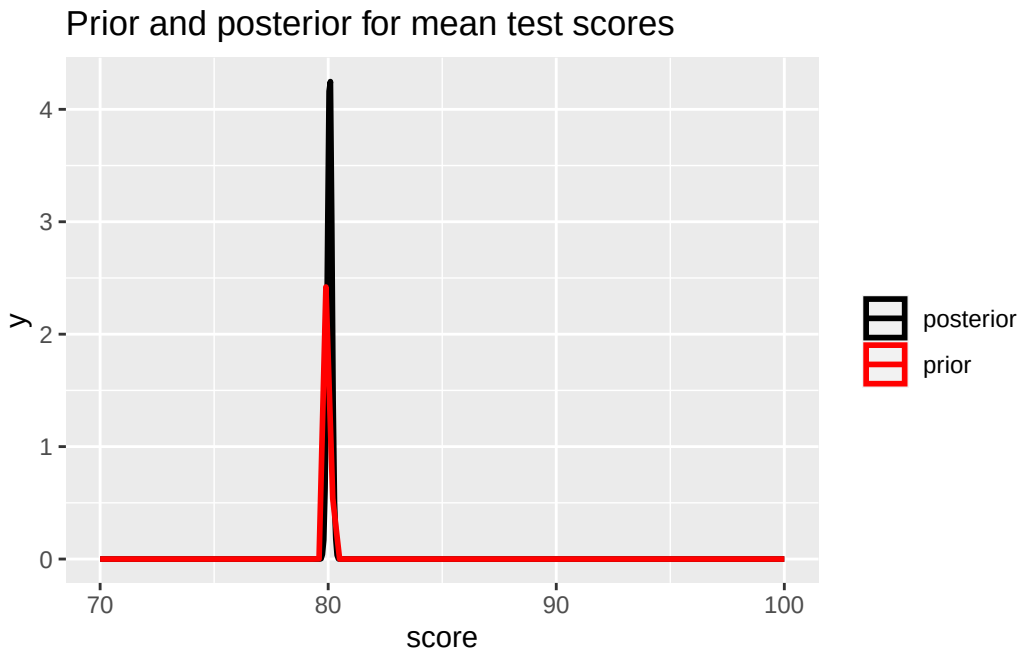
```
fit_informative
```

```
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       80.07    0.00 0.09    79.88    80.00    80.07    80.13    80.26   806
sigma    21.38    0.03 0.74    19.99    20.88    21.38    21.85    22.92   674
lp__  -1548.34    0.05 0.97 -1550.93 -1548.73 -1548.02 -1547.66 -1547.38   337
      Rhat
mu       1
sigma    1
lp__     1

Samples were drawn using NUTS(diag_e) at Sun Feb 12 12:57:07 2023.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

The estimates have changed slightly. Now we have a mean μ parameter of 80.06 rather than 86.73. This is significantly lower. In addition, the estimate for σ is 21.42 instead of 20.40. In general, the standard error has decreased. This could be because our prior was more confident, and this in turn influences the final posterior. Our credible intervals are also different as a consequence.

```
dsamples2 <- fit_informative  |>
  gather_draws(mu, sigma) # gather = long format tibble
```

```
dsamples2 |>
  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")
```



The prior for the mean test score is much more confident than it was before, and so is the posterior. That is, the data is reinforcing the bias that we are encoding in our model. Both have a very narrow and steep bell-curve.

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 b/c K = 1 in stan
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)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 3.1e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.31 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: 0.073 seconds (Warm-up)
Chain 1:                0.042 seconds (Sampling)
Chain 1:                0.115 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: 0.089 seconds (Warm-up)
Chain 2:                0.045 seconds (Sampling)
Chain 2:                0.134 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: 0.074 seconds (Warm-up)
Chain 3:                0.046 seconds (Sampling)
Chain 3:                0.12 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 1.5e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.15 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: 0.098 seconds (Warm-up)
Chain 4:                0.049 seconds (Sampling)
Chain 4:                0.147 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?

Building a linear model:

```
lin_mod <- lm(kid_score ~ mom_hs, data = kidiq)
summary(lin_mod)
```

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

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 ***
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
```

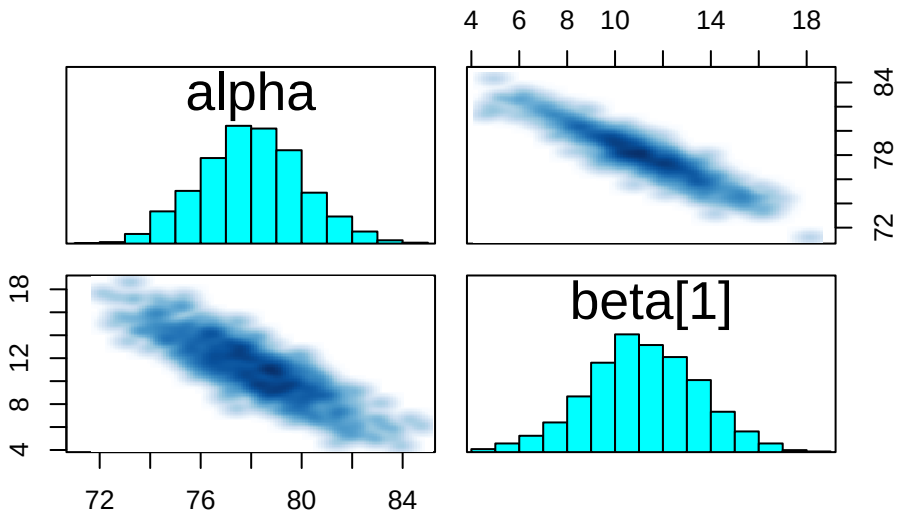
Now look at what we got before:

```
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      78.02    0.08 2.00    74.21    76.66    78.01    79.35    82.10
beta[1]    11.14    0.09 2.31     6.45     9.63    11.10    12.70    15.59
sigma      19.81    0.02 0.70    18.53    19.33    19.78    20.25    21.26
lp__    -1514.42    0.05 1.26 -1517.67 -1514.96 -1514.10 -1513.50 -1512.96
        n_eff Rhat
alpha     701    1
beta[1]   675    1
sigma    1068    1
lp__      677    1

Samples were drawn using NUTS(diag_e) at Sun Feb 12 12:57:44 2023.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Firstly, we see that these estimates are very similar. This is interesting, since one is in a Bayesian framework. But look how low the standard error of the posterior distribution is in our Bayesian regression model. It is very small. In general, as we increase the number of data points (n), Bayesian inference becomes more influenced by the data, and our posterior distribution becomes less variable. This results in a distribution strongly centered around some point estimate. Typically, this point estimate is similar to the original MLE approach to finding coefficients. Thus, since we have lots of data this is not too surprising a result.

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



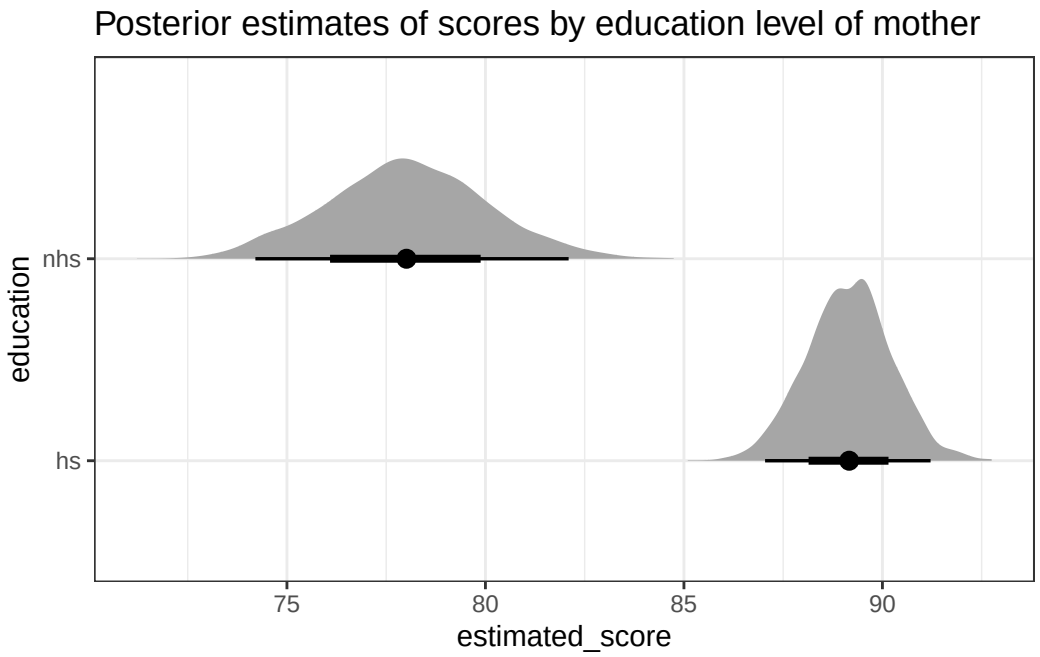
We see a strong negative correlation between the slope and intercept. This makes sense. We did not center our data, so the relationship between slope and intercept will be strengthened. As we increase the intercept, the slope will have to decrease to accurately fit the data. This is not good, as you can imagine that MCMC has less directions to move from where it samples (and it cannot take as large gradient steps to reach the optimal value). So it will make sampling from the distribution more difficult and slow convergence. In addition, it might mean that interpretation is more difficult. This correlation means that the intercept reflects a lack of any IQ for the mother - something that is impossible.

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.

```
X <- kidiq[,c('mom_hs', 'mom_iq')]
X$mom_iq <- X$mom_iq - mean(X$mom_iq) # Mean-Center
X <- as.matrix(X, ncol = 2)
K <- 2

data <- list(y = y, N = length(y),
             X = X, K = K)
```

```r
fit3 <- stan(file = here("code/models/kids3.stan"),
             data = data,
             iter = 1000)
```

```
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 / 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: 0.098 seconds (Warm-up)
Chain 1:                0.061 seconds (Sampling)
Chain 1:                0.159 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.2e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.12 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: 0.1 seconds (Warm-up)
Chain 2:                0.059 seconds (Sampling)
Chain 2:                0.159 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 / 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: 0.083 seconds (Warm-up)
Chain 3:                0.06 seconds (Sampling)
Chain 3:                0.143 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 1.2e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.12 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: 0.098 seconds (Warm-up)
Chain 4:                0.063 seconds (Sampling)
Chain 4:                0.161 seconds (Total)
Chain 4:
```

```
fit3
```

```
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     82.34    0.06 1.93    78.55    80.99    82.31    83.63    86.06
beta[1]    5.68    0.08 2.20     1.47     4.15     5.68     7.21     9.97
beta[2]    0.56    0.00 0.06     0.44     0.52     0.56     0.60     0.69
sigma     18.13    0.02 0.61    17.00    17.71    18.13    18.53    19.37
lp__    -1474.49    0.05 1.42 -1477.98 -1475.23 -1474.17 -1473.45 -1472.69
        n_eff Rhat
alpha     883    1
beta[1]   859    1
beta[2]  1449    1
sigma    1590    1
lp__      845    1

Samples were drawn using NUTS(diag_e) at Sun Feb 12 12:57:46 2023.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
```

convergence, Rhat=1).

The Mom's IQ has a coefficient of 0.57. This means that for a one-unit increase in IQ, there is a 0.57 increase in the kid's test score. This is the same regardless of the fact that we mean-centered. However, the intercept now reflects the baseline test score when the mother has a mean IQ. This baseline test score has expected value 82.38, given the mother has the average IQ.

Question 5

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

```
X <- kidiq[,c('kid_score', 'mom_hs', 'mom_iq')]
X$mom_iq <- X$mom_iq - mean(X$mom_iq)
lm_check <- lm(kid_score ~ mom_hs + mom_iq, data = X)
summary(lm_check)
```

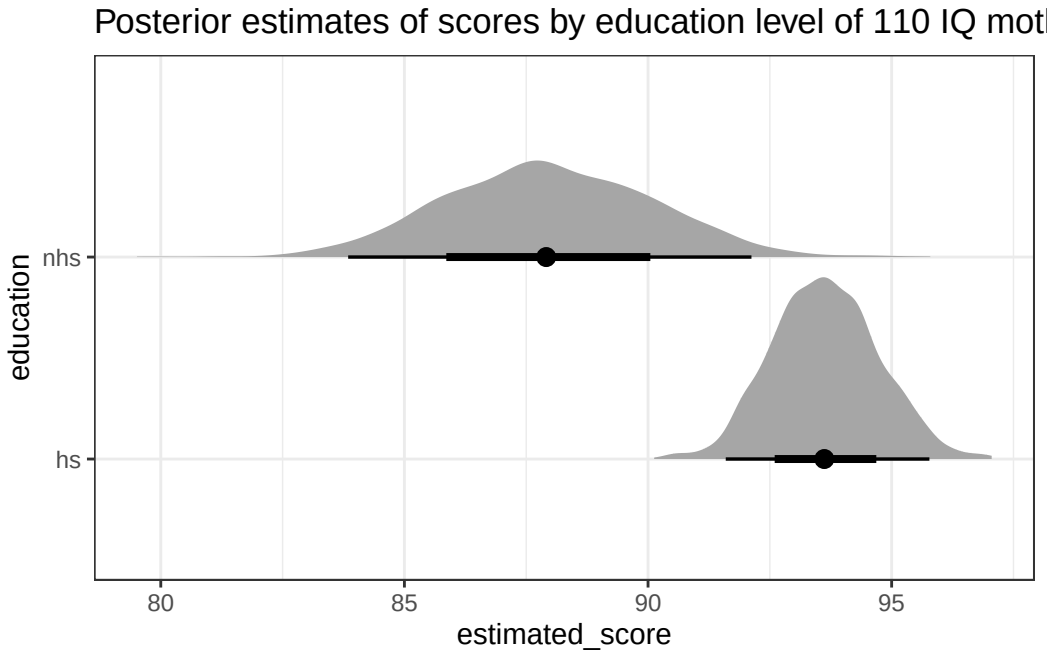
```
Call:
lm(formula = kid_score ~ mom_hs + mom_iq, data = X)

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 ***
mom_hs       5.95012    2.21181   2.690  0.00742 **
mom_iq       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
```

Co-efficient results are again very similar! Of course, since MCMC takes a sample, there is always an element of stochasticity to this so our coefficients will not be exactly the same. But they are exceptionally close.

Question 6

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

```
fit3 |>
  spread_draws(alpha, beta[k], sigma) |>
  pivot_wider(names_from = k, names_prefix ='beta', values_from = 'beta') %>%
     mutate(nhs = alpha + beta2*(110- mean(kidiq$mom_iq)),
          hs = alpha + beta1 + beta2*(110- mean(kidiq$mom_iq))) |>
  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 110 IQ mother ")
```



Question 7

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

```r
post_samples <- extract(fit3)

sigma <- post_samples[["sigma"]]
lin_pred <- post_samples[["alpha"]] +
  post_samples[["beta"]][,1]*1 +
  post_samples[["beta"]][,2]*(95 - mean(kidiq$mom_iq))
y_new <- rnorm(n = length(sigma),mean = lin_pred, sd = sigma)
```

```r
res <- data.frame(y=y_new)
ggplot(res, aes(x=y)) + geom_histogram(bins = 30) +
labs(x="Estimated Test Score",
y="Frequency",
title="Posterior Prediction Distribution",
subtitle="Distribution of Test Scores for a kid who has a mother that graduated high schoo
```