Week 5: Bayesian linear regression and introduction to Stan

11/02/23

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.

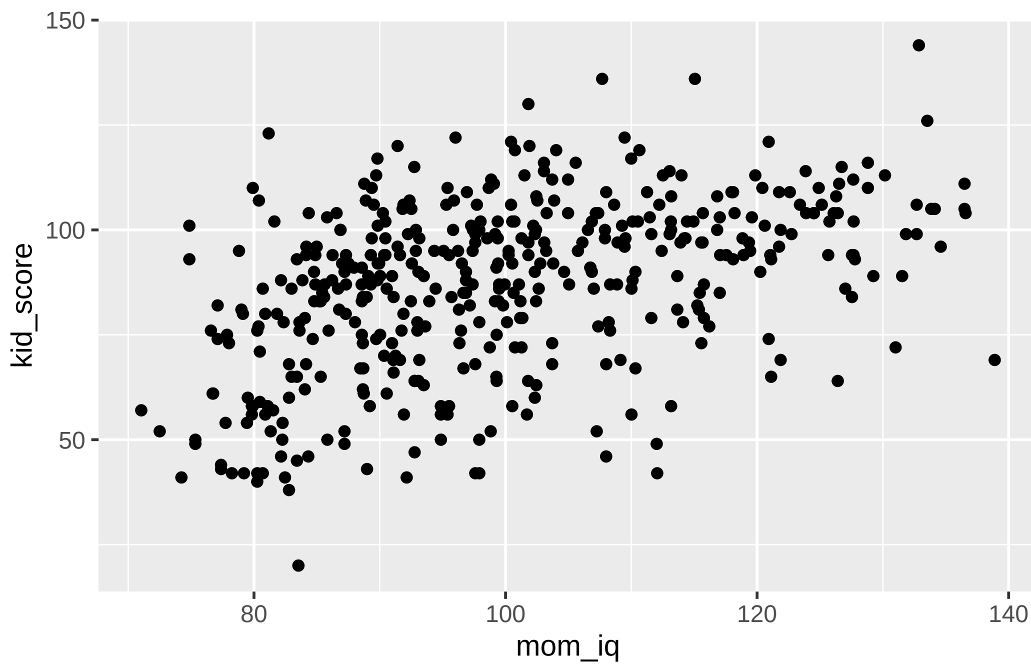
Descriptives

Question 1

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

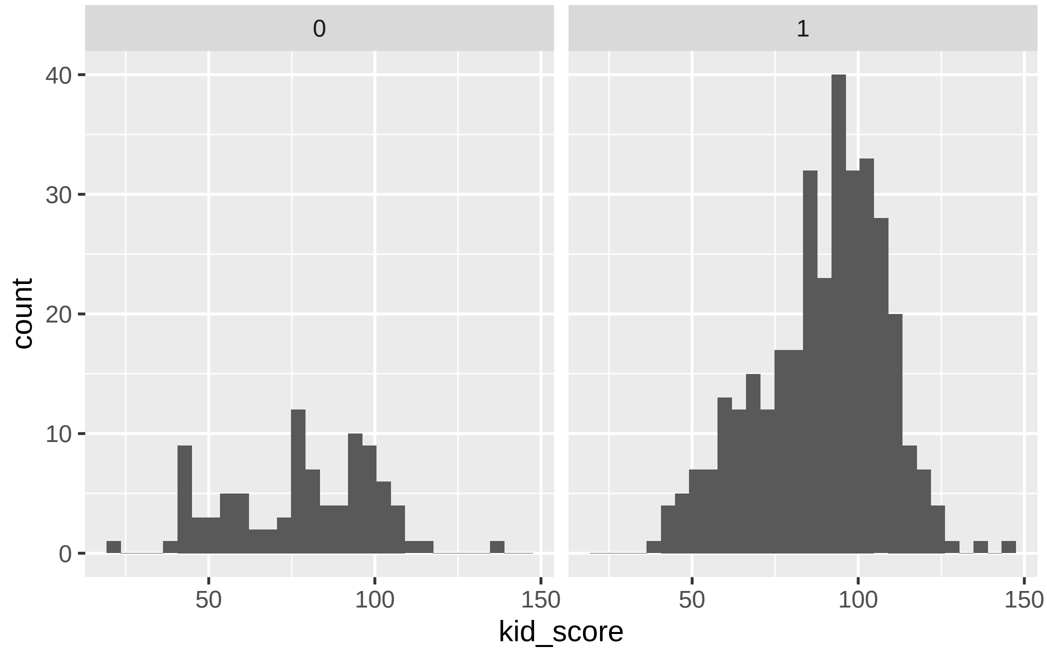
- Explain what your graph/ tables show
- Choose a graph type that's appropriate to the data type

```
ggplot(data=kidiq)+
  geom_point(aes(x=mom_iq, y=kid_score))
```

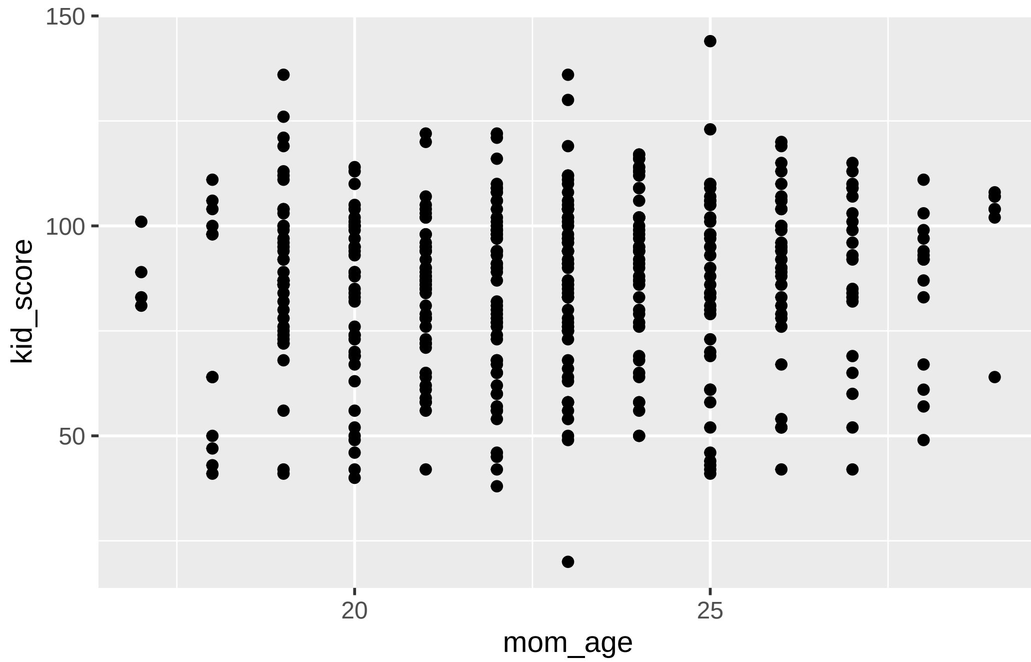



```
ggplot(data=kidiq)+
  geom_histogram(aes(kid_score))+
```

```
  facet_grid(~mom_hs)
```




```
ggplot(data=kidiq)+
  geom_point(aes(x=mom_age, y=kid_score))
```



1. Since both kid_score and mom_iq are numerical, I first plot a scatter plot of kid's IQ score vs mom's IQ score, and found that kid's IQ score increases as mom's IQ score increases.

2. Since mom_hs is binary and kid_score is numerical, I then plot histograms of kid's IQ by mom's high school status, and found that for mother with a high school degree, the kid's IQ has larger density on higher IQ scores than for mother without a high school degree.

3. Since both kid_score and mom_age are numerical, I also plot a scatter plot of kid's IQ score vs mom's age, and found that there is no clear pattern between kid's IQ and mom's age.

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.

```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:

```r
fit <- stan(file = here("code/models/kids2.stan"),
            data = data,
            chains = 3,
            iter = 500,
            seed = 1)
```

```
SAMPLING FOR MODEL 'kids2' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 2.9e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.29 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: 0.014204 seconds (Warm-up)
Chain 1:                0.007131 seconds (Sampling)
Chain 1:                0.021335 seconds (Total)
```

```
Chain 1:

SAMPLING FOR MODEL 'kids2' 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 / 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: 0.011218 seconds (Warm-up)
Chain 2:                0.007308 seconds (Sampling)
Chain 2:                0.018526 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'kids2' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 8e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 3: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 3: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 3: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 3: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 3: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 3: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 3: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 3: Iteration: 350 / 500 [ 70%]  (Sampling)
```

```
Chain 3: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 3: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 3: Iteration: 500 / 500 [100%]  (Sampling)
Chain 3: 
Chain 3:  Elapsed Time: 0.011761 seconds (Warm-up)
Chain 3:                0.007271 seconds (Sampling)
Chain 3:                0.019032 seconds (Total)
Chain 3: 
```

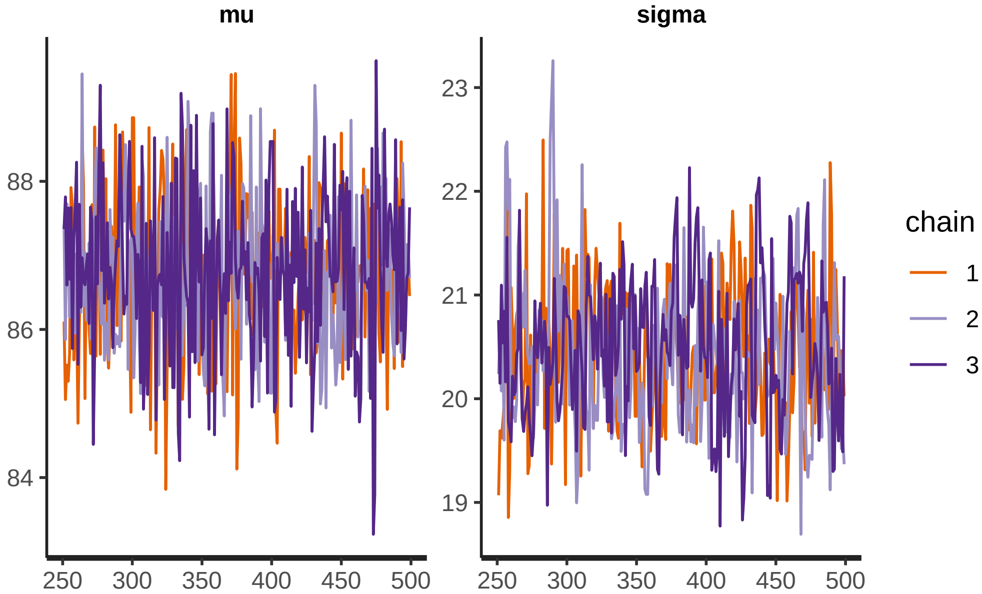
Look at the summary

```
fit
```

```
Inference for Stan model: kids2.
3 chains, each with iter=500; warmup=250; thin=1; 
post-warmup draws per chain=250, total post-warmup draws=750.

          mean se_mean   sd     2.5%      25%      50%      75%    97.5% n_eff
mu       86.79    0.04 1.04    84.81    86.07    86.80    87.48    88.77   743
sigma    20.45    0.04 0.69    19.17    19.97    20.44    20.92    21.87   315
lp__  -1525.83    0.06 1.07 -1528.49 -1526.26 -1525.51 -1525.05 -1524.78   369
      Rhat
mu    1.00
sigma 1.00
lp__  1.02

Samples were drawn using NUTS(diag_e) at Sat Feb 11 19:47:56 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).
```

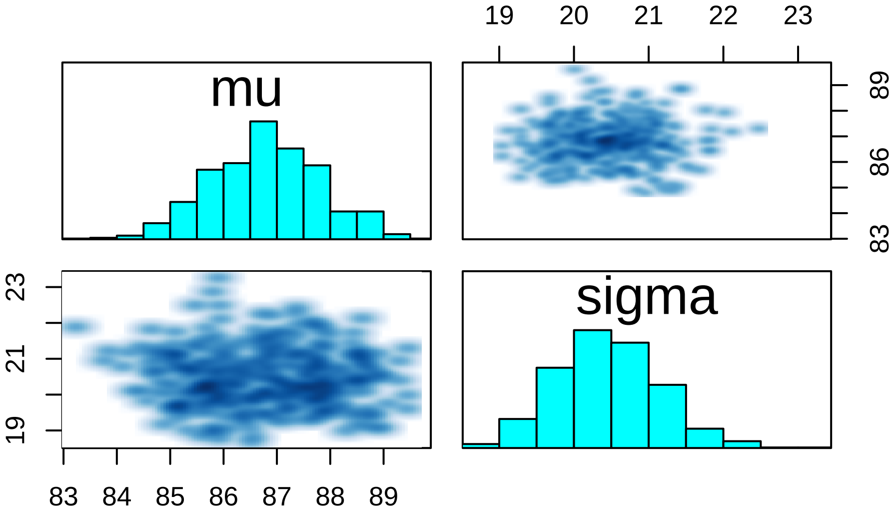
Traceplot

```
traceplot(fit)
```

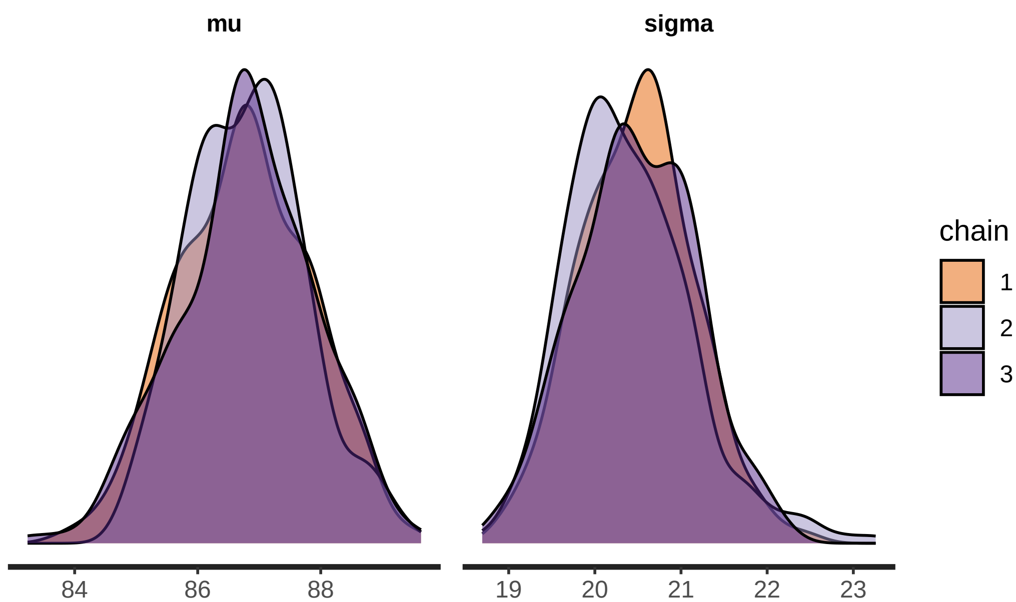


All looks fine.

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




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



Understanding output

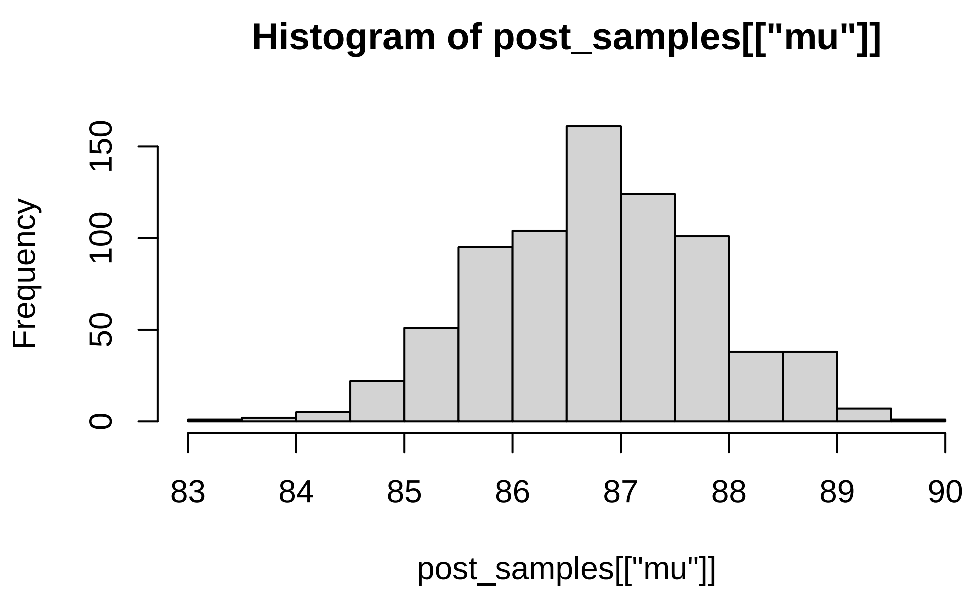
What does the model actually give us? A number of samples from the posteriors. To see this, we can use `extract` to get the samples.

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

```
[1] 86.52759 85.49155 86.30489 86.47727 86.79935 86.86390
```

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] 86.79872
```

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

```
    2.5%
84.81158
```

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

```
   97.5%
88.76504
```

Plot estimates

There are a bunch of packages, built-in functions that let you plot the estimates from the model, and I encourage you to explore these options (particularly in `bayesplot`, which we will most likely be using later on). I like using the `tidybayes` package, which allows us to easily get the posterior samples in a tidy format (e.g. using gather draws to get in long format). Once we have that, it's easy to just pipe and do ggplots as usual.

Get the posterior samples for mu and sigma in long format:

```
dsamples <- fit  |>
  gather_draws(mu, sigma) # gather = long format
dsamples
```

```
# A tibble: 1,500 x 5
# Groups:   .variable [2]
   .chain .iteration .draw .variable .value
    <int>      <int> <int> <chr>      <dbl>
 1      1          1     1 mu          86.1
 2      1          2     2 mu          85.1
 3      1          3     3 mu          85.5
 4      1          4     4 mu          85.3
 5      1          5     5 mu          85.7
 6      1          6     6 mu          87.9
 7      1          7     7 mu          87.7
 8      1          8     8 mu          85.6
 9      1          9     9 mu          86.0
10      1         10    10 mu          86.2
# ... with 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  86.1  19.1
 2      1          2     2  85.1  19.7
 3      1          3     3  85.5  19.6
 4      1          4     4  85.3  19.8
 5      1          5     5  85.7  19.9
 6      1          6     6  87.9  21.4
 7      1          7     7  87.7  22.1
 8      1          8     8  85.6  18.9
 9      1          9     9  86.0  19.3
10      1         10    10  86.2  21.1
# ... with 740 more rows
```

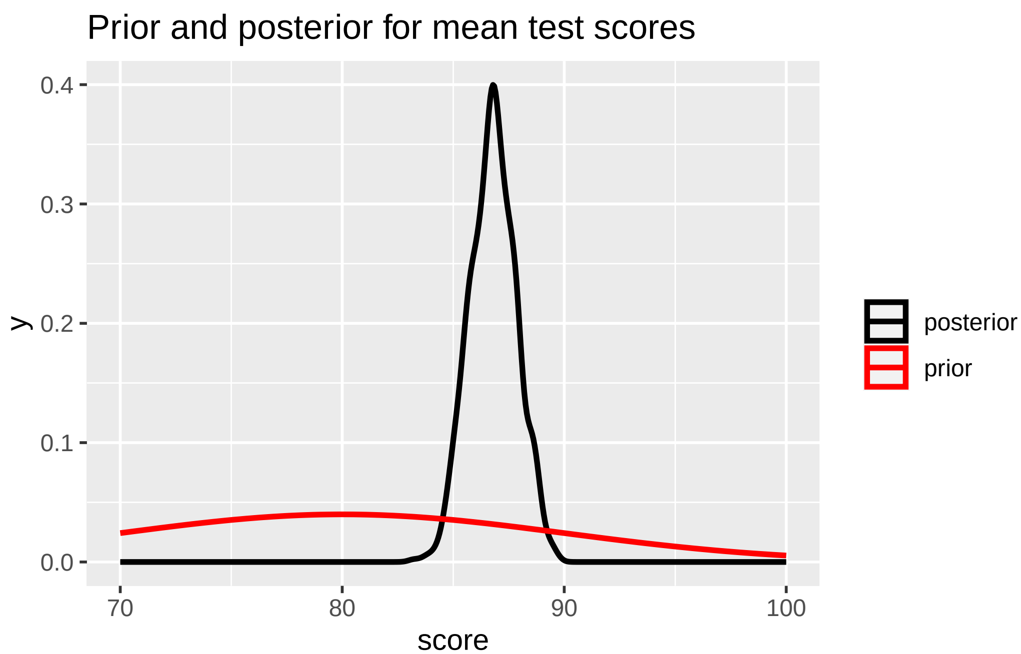
```r
# quickly calculate the quantiles using

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

```
# A tibble: 2 x 7
  .variable .value .lower .upper .width .point .interval
  <chr>      <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
1 mu          86.8   85.5   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")
```



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
sigma0 <- 0.1

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

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

```r
              iter = 500,
              seed = 1)
```

```
SAMPLING FOR MODEL 'kids2' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 1.4e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.14 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: 0.009571 seconds (Warm-up)
Chain 1:                0.008127 seconds (Sampling)
Chain 1:                0.017698 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'kids2' 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 / 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)
```

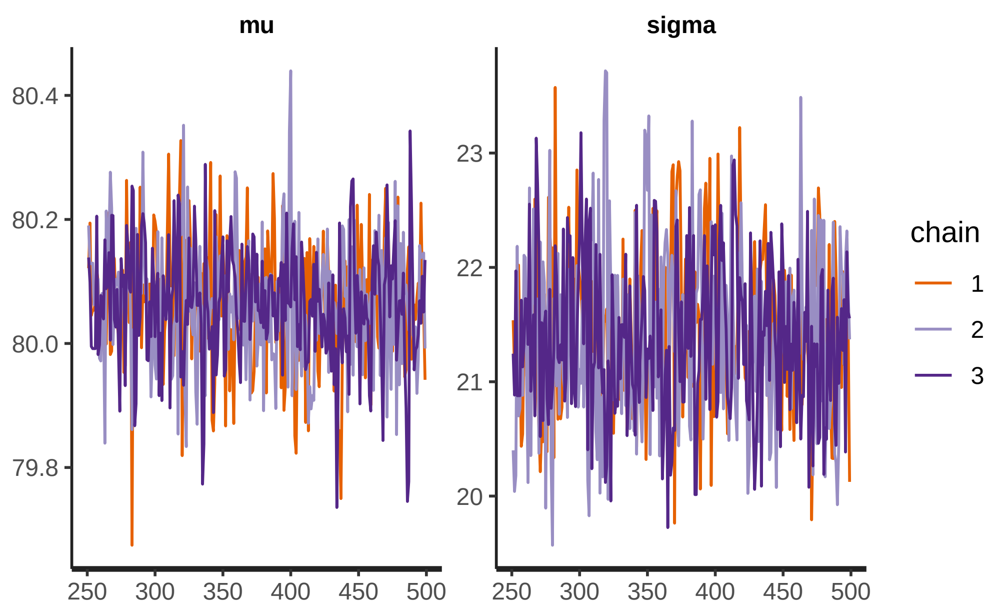
```
Chain 2: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 2: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 2: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 2: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 2: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 2: Iteration: 500 / 500 [100%]  (Sampling)
Chain 2: 
Chain 2:  Elapsed Time: 0.008712 seconds (Warm-up)
Chain 2:                0.007213 seconds (Sampling)
Chain 2:                0.015925 seconds (Total)
Chain 2: 

SAMPLING FOR MODEL 'kids2' 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 / 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: 0.009109 seconds (Warm-up)
Chain 3:                0.008079 seconds (Sampling)
Chain 3:                0.017188 seconds (Total)
Chain 3: 
```

```
fit
```

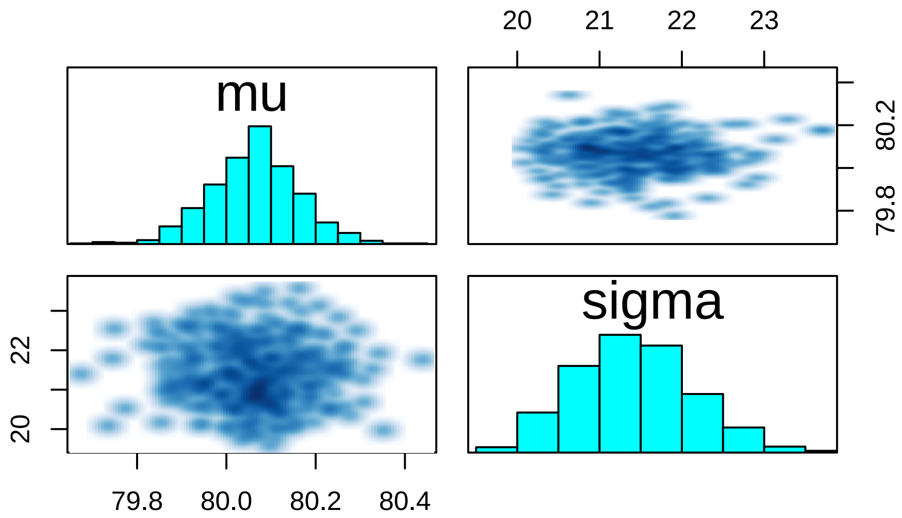
```
Inference for Stan model: kids2.
3 chains, each with iter=500; warmup=250; thin=1; 
post-warmup draws per chain=250, total post-warmup draws=750.
```

```
          mean se_mean   sd     2.5%      25%      50%      75%    97.5% n_eff
mu       80.06    0.00 0.10    79.86    80.00    80.07    80.12    80.25   669
sigma    21.42    0.03 0.72    20.09    20.90    21.40    21.90    22.90   618
lp__  -1548.35    0.05 1.03 -1551.09 -1548.67 -1548.04 -1547.67 -1547.40   352
      Rhat
mu       1
sigma    1
lp__     1

Samples were drawn using NUTS(diag_e) at Sat Feb 11 19:47:59 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).
```

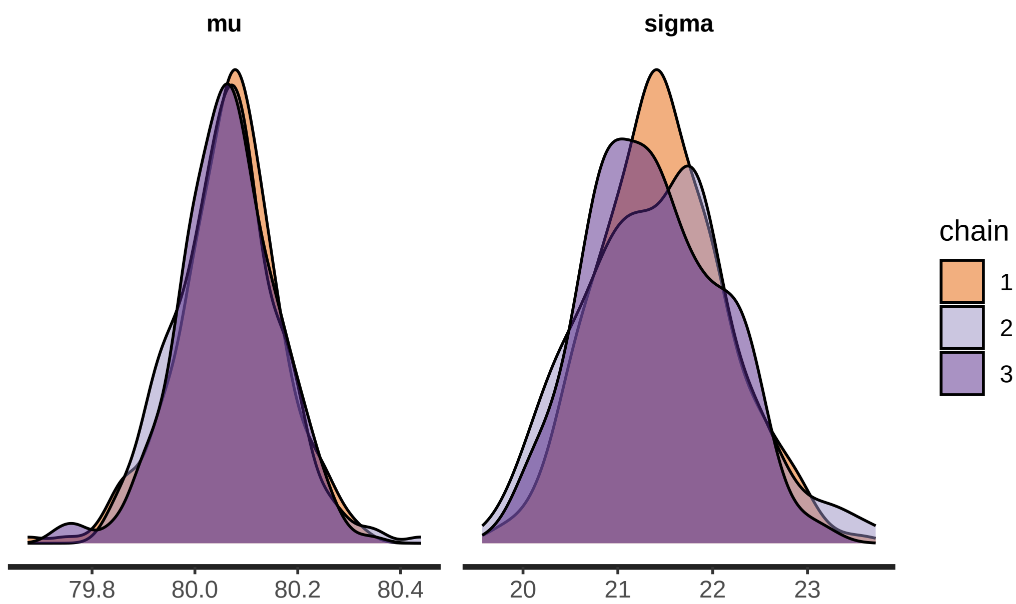
```r
traceplot(fit)
```




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



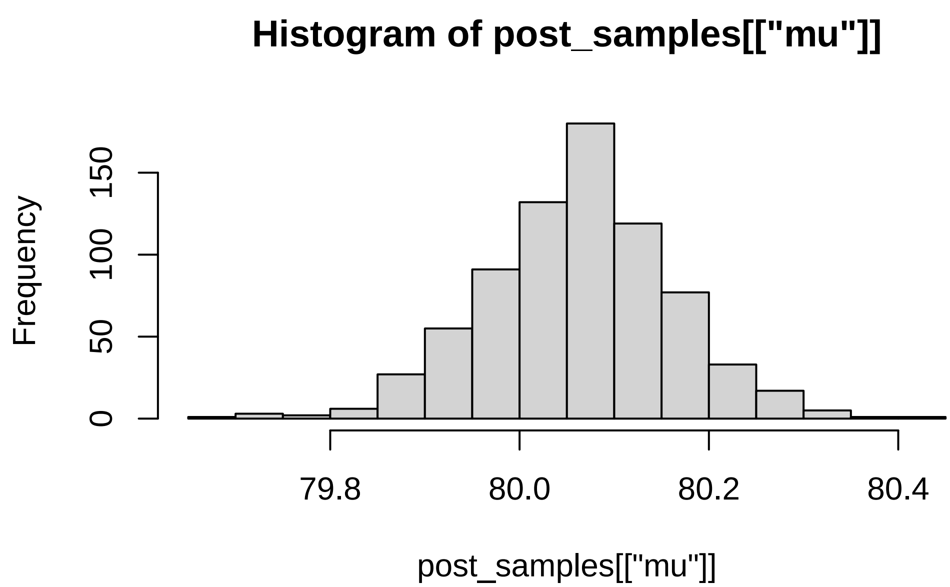

```r
stan_dens(fit, separate_chains = TRUE)
```



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

```
[1] 80.04107 80.06757 80.09096 80.19419 80.29191 79.67466
```

```r
hist(post_samples[["mu"]])
```




```r
median(post_samples[["mu"]])
```

```
[1] 80.06706
```

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

```
    2.5%
79.86062
```

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

```
   97.5%
80.25431
```

```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          80.1
 2      1          2     2 mu          80.2
 3      1          3     3 mu          80.0
 4      1          4     4 mu          80.1
 5      1          5     5 mu          80.1
 6      1          6     6 mu          80.1
 7      1          7     7 mu          80.1
 8      1          8     8 mu          80.1
 9      1          9     9 mu          80.0
10      1         10    10 mu          80.1
# ... with 1,490 more rows
```

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

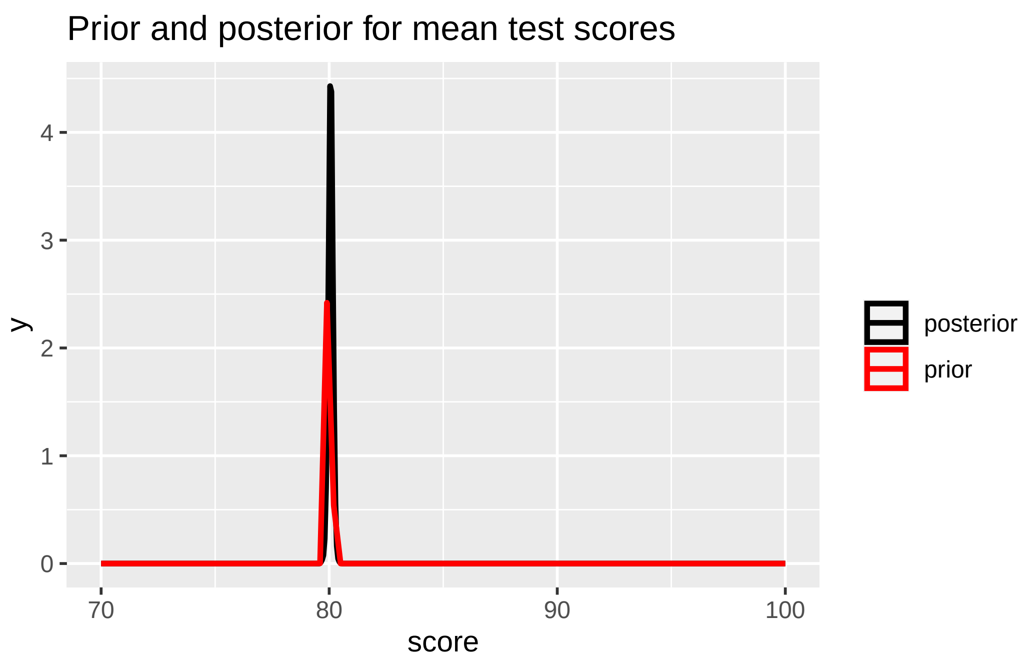
```
# A tibble: 750 x 5
   .chain .iteration .draw    mu sigma
    <int>      <int> <int> <dbl> <dbl>
 1      1          1     1  80.1  21.5
 2      1          2     2  80.2  21.1
 3      1          3     3  80.0  21.0
 4      1          4     4  80.1  22.0
 5      1          5     5  80.1  22.0
 6      1          6     6  80.1  21.3
 7      1          7     7  80.1  20.4
 8      1          8     8  80.1  20.5
```

```
 9       1           9      9   80.0   21.4
10       1          10     10   80.1   21.7
# ... 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          80.1   79.9   80.2    0.8 median qi
2 sigma       21.4   20.5   22.4    0.8 median qi
```

```
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")
```



The estimates changed. Previously, the median of mu is 86.8 and median of sigma is 20.4, but now the median of mu is 80.1 and median of sigma is 21.4.

Adding covariates

Now let's see how kid's test scores are related to mother's education. We want to run the simple linear regression

$$Score = \alpha + \beta X$$

where $X = 1$ if the mother finished high school and zero otherwise.

`kid3.stan` has the stan model to do this. Notice now we have some inputs related to the design matrix X and the number of covariates (in this case, it's just 1).

Let's get the data we need and run the model.

```
X <- as.matrix(kidiq$mom_hs, ncol = 1) # force this to be a matrix
K <- 1

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

```r
fit2 <- stan(file = here("code/models/kids3.stan"),
             data = data,
             iter = 1000,
             seed = 1)
```

```
SAMPLING FOR MODEL 'kids3' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 7.5e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.75 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: 0.183939 seconds (Warm-up)
Chain 1:                0.134833 seconds (Sampling)
Chain 1:                0.318772 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'kids3' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 3.7e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.37 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: 0.188002 seconds (Warm-up)
Chain 2:                0.120174 seconds (Sampling)
Chain 2:                0.308176 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'kids3' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 3.3e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.33 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: 0.231008 seconds (Warm-up)
Chain 3:                0.13619 seconds (Sampling)
Chain 3:                0.367198 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'kids3' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 3.8e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.38 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: 0.182679 seconds (Warm-up)
Chain 4:                0.127646 seconds (Sampling)
Chain 4:                0.310325 seconds (Total)
Chain 4: 
```

Question 3

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

```r
fit2
```

```
Inference for Stan model: kids3.
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      78.00    0.07 1.94    74.12    76.74    78.06    79.33    81.67
beta[1]    11.18    0.08 2.19     7.00     9.66    11.17    12.60    15.56
sigma      19.83    0.02 0.66    18.59    19.39    19.81    20.25    21.23
lp__    -1514.33    0.04 1.21 -1517.35 -1514.93 -1514.03 -1513.44 -1512.98
        n_eff Rhat
alpha     804 1.01
beta[1]   846 1.01
sigma    1042 1.00
```

```
lp__      758 1.00

Samples were drawn using NUTS(diag_e) at Sat Feb 11 19:48:34 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).
```

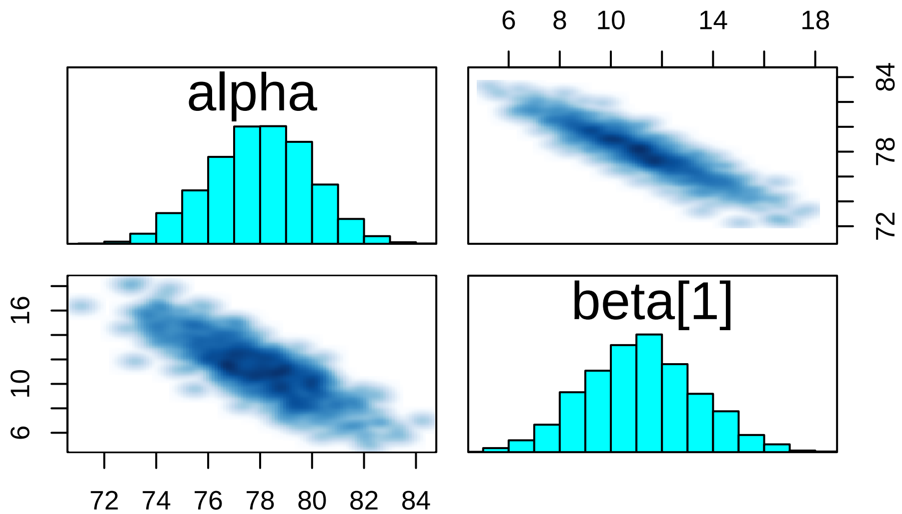
```r
model<-lm(kid_score~as.factor(mom_hs), data=kidiq)
summary(model)
```

```
Call:
lm(formula = kid_score ~ as.factor(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 ***
as.factor(mom_hs)1   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
```

We see that the stan estimates alpha and beta are very similar to the intercept and slope from the lm model.

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[1]"))
```



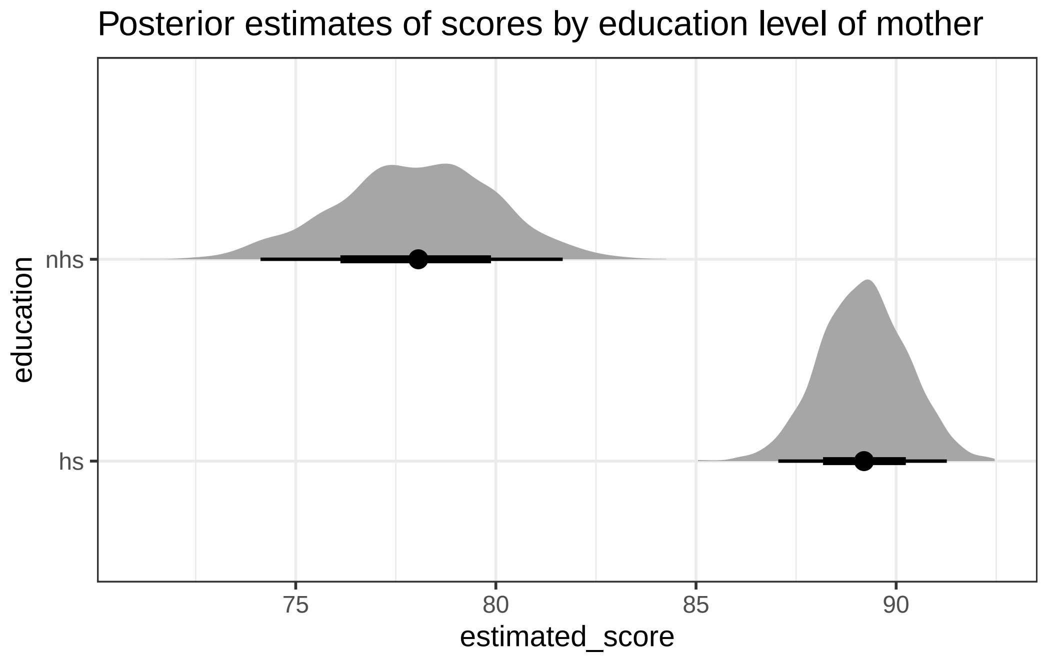
We see that if we sample high values of alpha, we are likely to sample low values of beta (which is expected when we fit a straight line on the data). This is a potential problem because we will get narrower results when sampling these parameters, which makes the sampling process inefficient.

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 <- cbind(as.matrix(kidiq$mom_hs), as.matrix(kidiq$mom_iq - mean(kidiq$mom_iq)))
K <- 2

data <- list(y = y, N = length(y),
             X =X, K = K)
fit3 <- stan(file = here("code/models/kids3.stan"),
             data = data,
             iter = 1000,
             seed = 1)
```

```
SAMPLING FOR MODEL 'kids3' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 3.8e-05 seconds
```

```
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.38 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: 0.214734 seconds (Warm-up)
Chain 1:                0.160744 seconds (Sampling)
Chain 1:                0.375478 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'kids3' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 3.2e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.32 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: 0.259357 seconds (Warm-up)
```

```
Chain 2:                0.148798 seconds (Sampling)
Chain 2:                0.408155 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'kids3' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 3.2e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.32 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: 0.252556 seconds (Warm-up)
Chain 3:                0.156238 seconds (Sampling)
Chain 3:                0.408794 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'kids3' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 3.5e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.35 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: 0.253768 seconds (Warm-up)
Chain 4:                0.149135 seconds (Sampling)
Chain 4:                0.402903 seconds (Total)
Chain 4: 
```

```
fit3
```

```
Inference for Stan model: kids3.
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      82.33    0.06 1.89    78.71    81.05    82.37    83.61    86.13
beta[1]     5.67    0.07 2.13     1.48     4.23     5.70     7.12     9.80
beta[2]     0.56    0.00 0.06     0.45     0.53     0.57     0.60     0.68
sigma      18.14    0.02 0.60    17.00    17.73    18.11    18.55    19.35
lp__    -1474.44    0.05 1.42 -1477.94 -1475.13 -1474.08 -1473.40 -1472.67
        n_eff Rhat
alpha     970 1.00
beta[1]   937 1.00
beta[2]  1280 1.00
sigma    1447 1.00
lp__      984 1.01

Samples were drawn using NUTS(diag_e) at Sat Feb 11 19:48:36 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).
```

Interpret the coefficient on the (centered) mum's IQ: For every one unit increase in centered mom's IQ, the expected kid's test score increases 0.56, holding all other variables (mom's high school degree status) constant.

Question 5

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

```r
kidiq<-kidiq|>
  mutate(mom_iq_center=mom_iq-mean(mom_iq))
summary(lm(kid_score~as.factor(mom_hs)+mom_iq_center,data=kidiq))
```

```

Call:
lm(formula = kid_score ~ as.factor(mom_hs) + mom_iq_center, data = kidiq)

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 ***
as.factor(mom_hs)1  5.95012    2.21181   2.690  0.00742 ** 
mom_iq_center       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
```

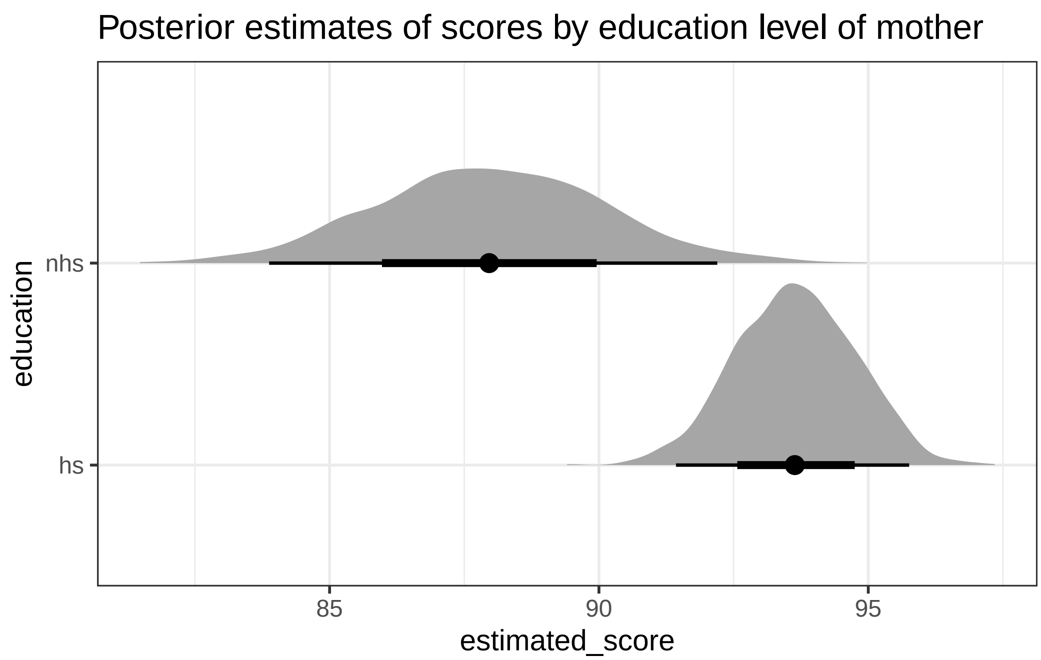
We see that stan's coefficient estimates are similar to the ones from lm model.

Question 6

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

```r
fit3 |>
  spread_draws(alpha, beta[k], sigma) |>
  pivot_wider(names_from = k, values_from = beta,  names_glue = "beta{k}") |>
  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() +
```

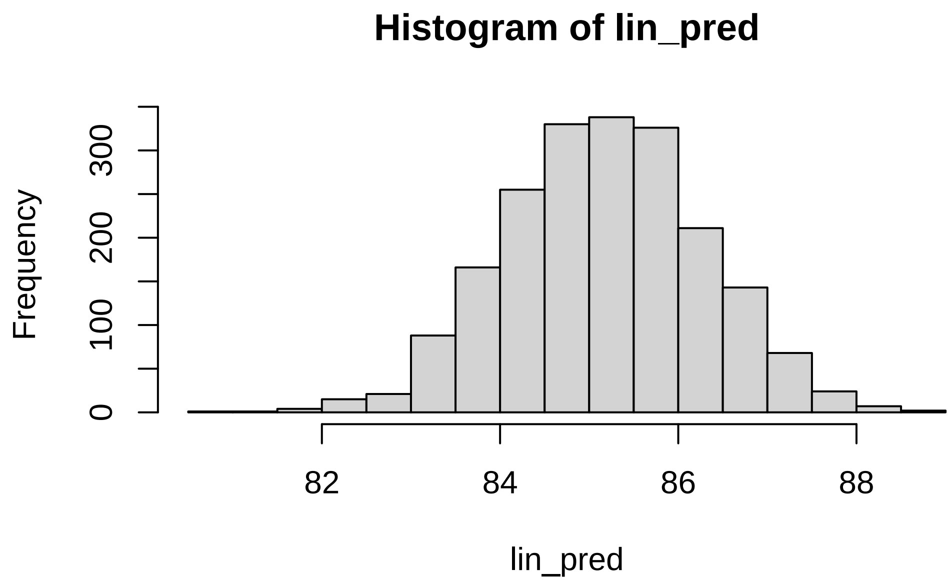
```r
  theme_bw() +
  ggtitle("Posterior estimates of scores by education level of mother")
```



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)
alpha <- post_samples[["alpha"]]
beta1 <- post_samples[["beta"]][,1]
beta2 <- post_samples[["beta"]][,2]

lin_pred <- alpha + beta1 + beta2*(95-mean(kidiq$mom_iq))
hist(lin_pred)
```


Histogram of lin_pred
Frequency
0
100
200
300
82
84
86
88
lin_pred