Week 5: Bayesian linear regression and introduction to Stan

Fred Haochen Song

14/02/23

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

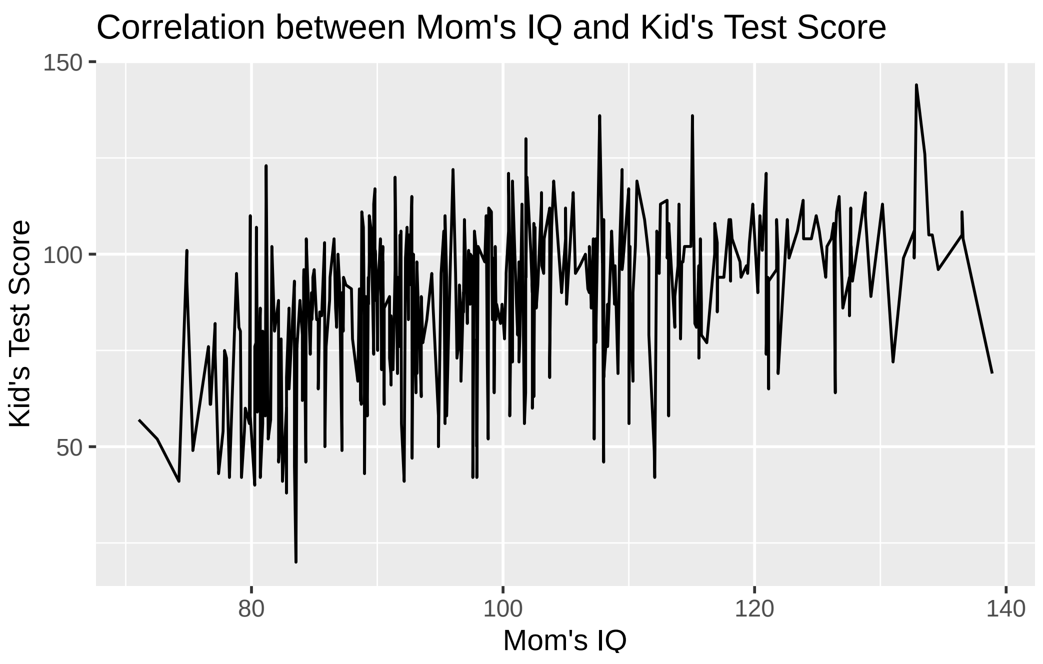
Question 1

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

Let's plot the first graph to be a trend between kid_score and mom_iq (the one that we have in the lecture slides):

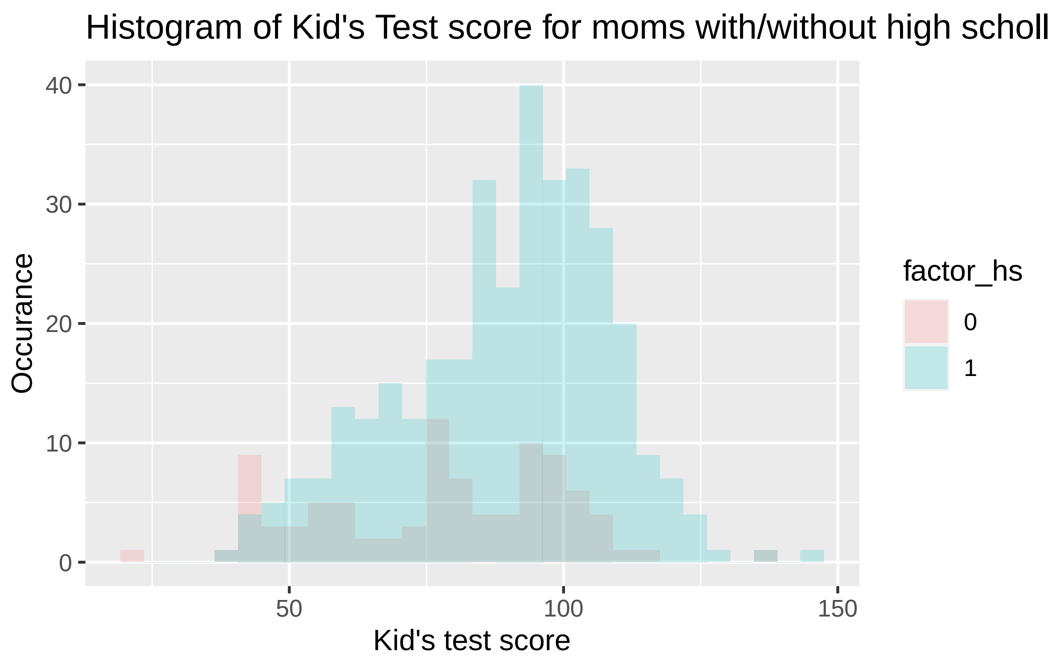
```r
kidiq |>
  ggplot(aes(x = mom_iq, y = kid_score)) +
  labs(title = "Correlation between Mom's IQ and Kid's Test Score",
       x = "Mom's IQ",
       y = "Kid's Test Score") +
  geom_line()
```



As we can see, there is in general a small trend that the higher the mom's iq, the higher the kid's test score will be.

I also want to investigate the general distribution between moms who have high school degree and moms who do not have high school degree, in the below histogram.

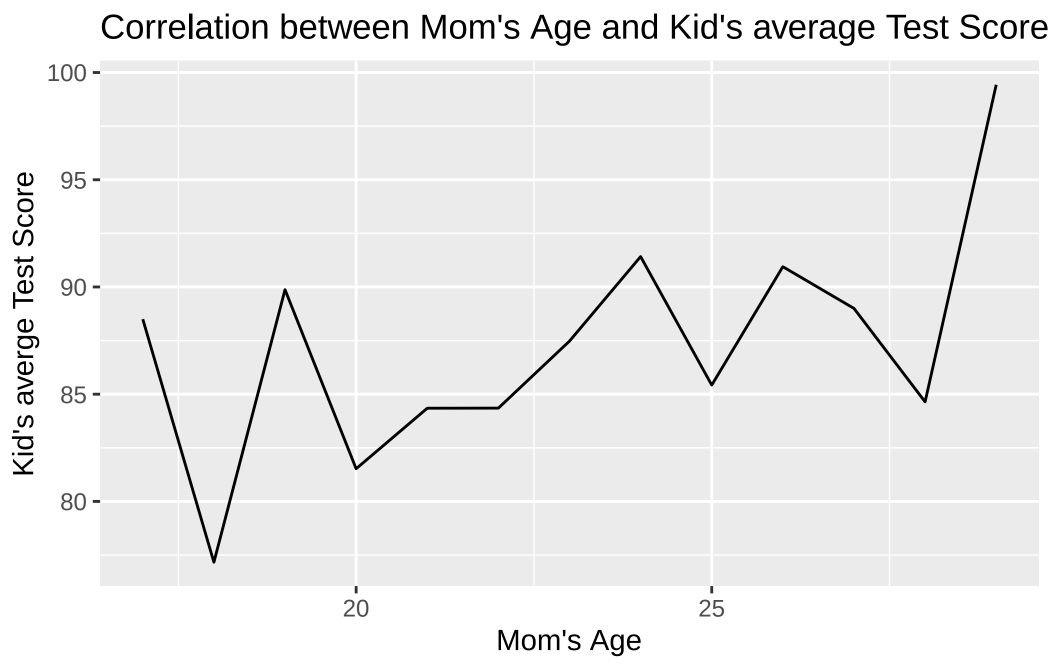
```r
kidiq |>
  mutate(factor_hs = as.factor(mom_hs)) |>
  ggplot(aes(x = kid_score, fill = factor_hs)) +
  labs(title = "Histogram of Kid's Test score for moms with/without high scholl education"
       x = "Kid's test score",
       y = "Occurance") +
  geom_histogram(position = "identity", alpha = 0.2, bins = 30)
```



As we can see several things from this histogram plot, the data we have has a lower frequency of moms without high school degree compared to those with high school degree. In general, the kid's score is densed at around 80-100, with higher max value for the group of mom with high school degree and lower minimum value for the group of mom without high school degree.

The next plot I want to investigate a bit more on the relationship between kid's average test score given mom's age:

```
kidiq |>
  group_by(mom_age) |>
  mutate(avg_score = mean(kid_score)) |>
  ggplot(aes(x = mom_age, y = avg_score)) +
  labs(title = "Correlation between Mom's Age and Kid's average Test Score",
       x = "Mom's Age",
       y = "Kid's averge Test Score") +
  geom_line()
```



As we can see, there is a bit of a positive trend in the kid's average test score with mom's age. This might be a good effect factor we could consider later when fitting the model.

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:

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

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -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/4.2-arm64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/R
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'kids2' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 8e-06 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.08 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: 0.006948 seconds (Warm-up)
Chain 1:                0.003216 seconds (Sampling)
Chain 1:                0.010164 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'kids2' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 3e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.03 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: 0.006403 seconds (Warm-up)
Chain 2:                0.003308 seconds (Sampling)
Chain 2:                0.009711 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'kids2' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 3e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.03 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: 0.00785 seconds (Warm-up)
Chain 3:                0.003732 seconds (Sampling)
Chain 3:                0.011582 seconds (Total)
Chain 3: 
```

Look at the summary

```
fit
```

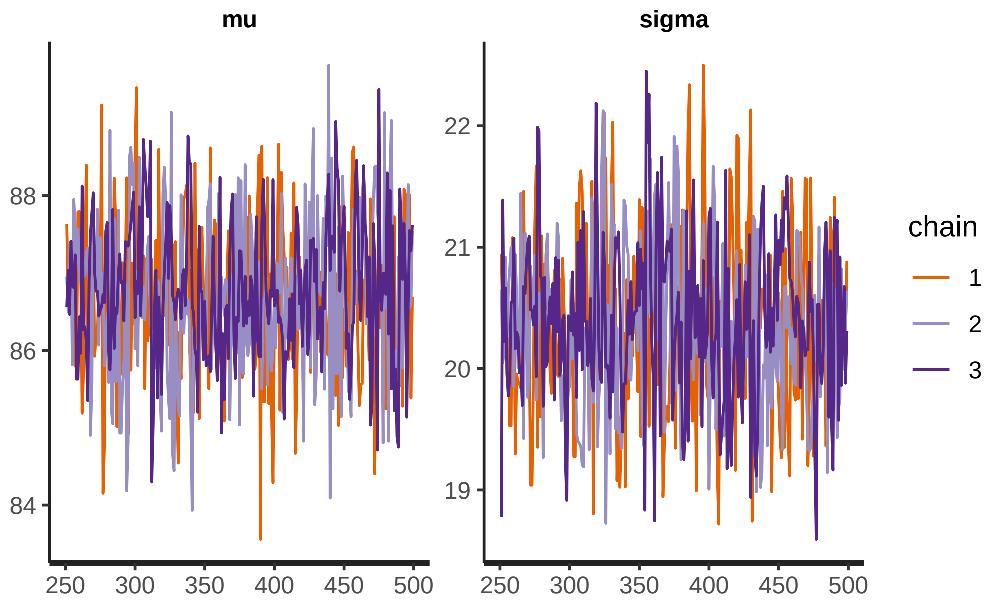
```
Inference for Stan model: kids2.
3 chains, each with iter=500; warmup=250; thin=1; 
post-warmup draws per chain=250, total post-warmup draws=750.

          mean se_mean   sd     2.5%      25%      50%      75%    97.5% n_eff
mu       86.72    0.04 0.98    84.87    86.03    86.73    87.38    88.57   642
sigma    20.37    0.03 0.69    19.04    19.89    20.34    20.85    21.73   571
lp__  -1525.77    0.06 0.99 -1528.59 -1526.24 -1525.45 -1525.05 -1524.79   269
      Rhat
mu       1
sigma    1
lp__     1

Samples were drawn using NUTS(diag_e) at Tue Feb 14 21:39:48 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).
```

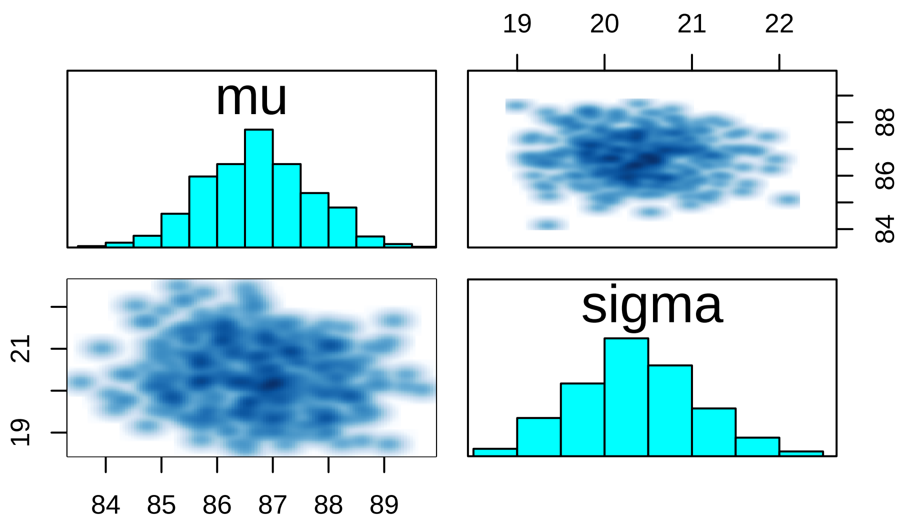
Traceplot

```
traceplot(fit)
```

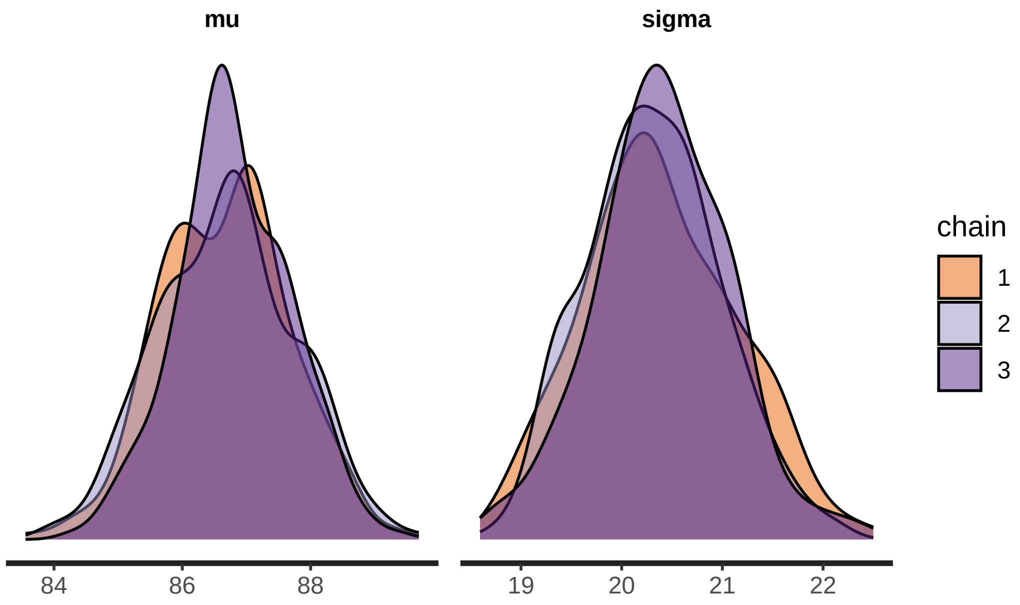


All looks fine. (densities)

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



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



Understanding output

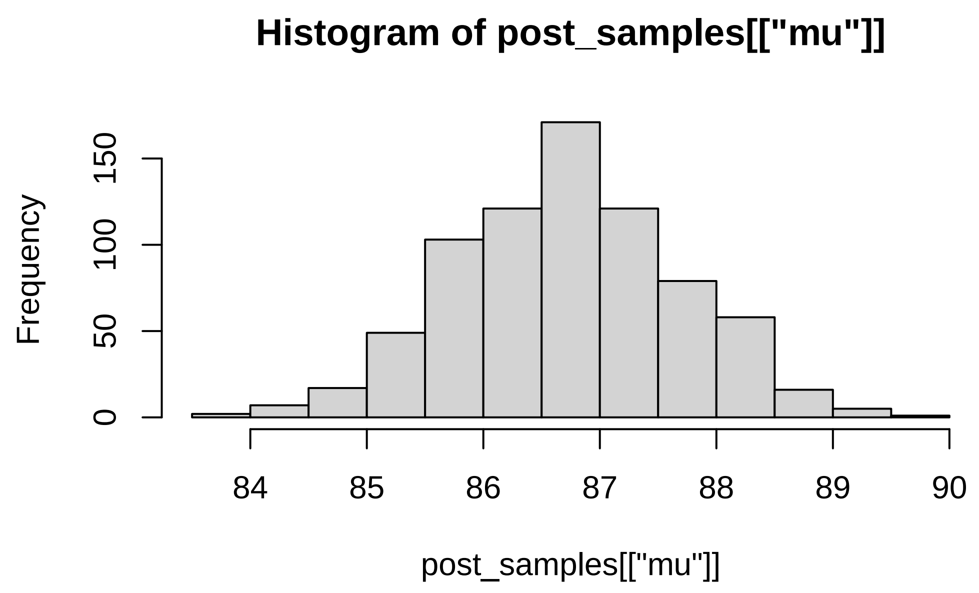
What does the model actually give us? A number of samples from the posteriors. To see this, we can use `extract` to get the samples.

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

```
[1] 85.88344 85.51950 87.63708 87.79372 87.21131 85.24509
```

This is a list, and in this case, each element of the list has 4000 samples. E.g. quickly plot a histogram of mu

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



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

```
[1] 86.73426
```

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

```
    2.5%
84.86945
```

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

```
   97.5%
88.57251
```

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 similar to pivot_longer
dsamples
```

```
# A tibble: 1,500 x 5
# Groups:   .variable [2]
   .chain .iteration .draw .variable .value
    <int>      <int> <int> <chr>      <dbl>
 1      1          1     1 mu          87.6
 2      1          2     2 mu          86.7
 3      1          3     3 mu          87.0
 4      1          4     4 mu          87.0
 5      1          5     5 mu          87.0
 6      1          6     6 mu          86.1
 7      1          7     7 mu          85.8
 8      1          8     8 mu          85.8
 9      1          9     9 mu          87.8
10      1         10    10 mu          87.8
# ... with 1,490 more rows
```

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

```
# A tibble: 750 x 5
   .chain .iteration .draw    mu sigma
    <int>      <int> <int> <dbl> <dbl>
 1      1          1     1  87.6  20.9
 2      1          2     2  86.7  20.4
 3      1          3     3  87.0  20.2
 4      1          4     4  87.0  20.2
 5       1           5     5  87.0  20.2
 6       1           6     6  86.1  20.2
 7       1           7     7  85.8  19.5
 8       1           8     8  85.8  19.5
 9       1           9     9  87.8  21.1
10       1          10    10  87.8  20.9
# ... with 740 more rows
```

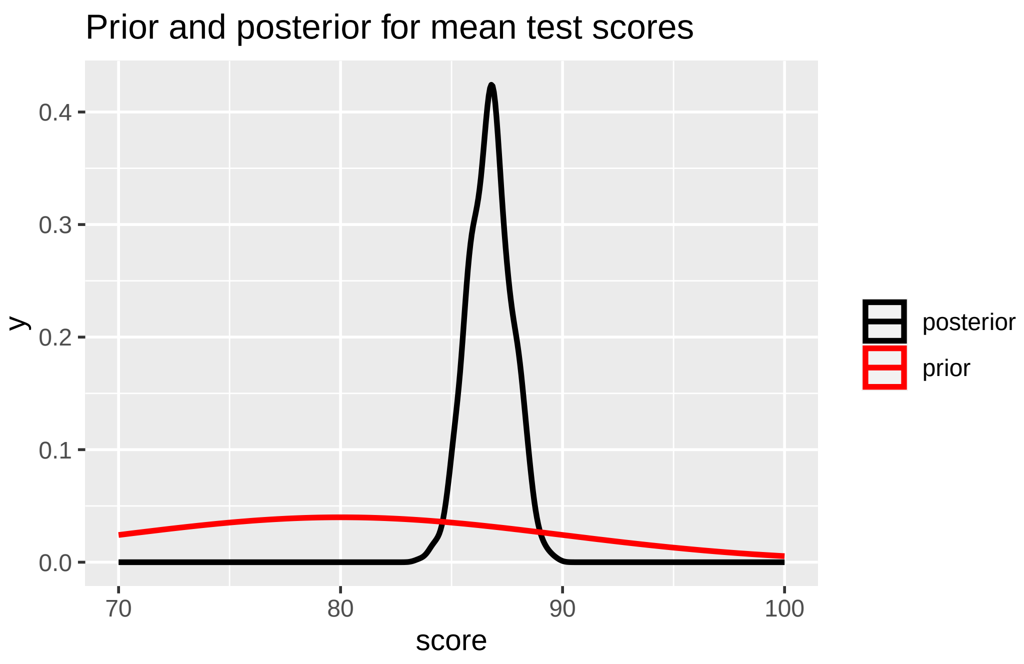
```
# quickly calculate the quantiles using

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          86.7   85.5   88.0    0.8 median qi
2 sigma       20.3   19.4   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.

changing them here:

```r
y <- kidiq$kid_score
mu0 <- 80
sigma0 <-0.1

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

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

```
SAMPLING FOR MODEL 'kids2' NOW (CHAIN 1).
```

```
Chain 1:
Chain 1: Gradient evaluation took 8e-06 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.08 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: 0.004093 seconds (Warm-up)
Chain 1:                0.003114 seconds (Sampling)
Chain 1:                0.007207 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'kids2' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 4e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 2: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 2: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 2: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 2: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 2: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 2: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 2: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 2: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 2: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 2: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 2: Iteration: 500 / 500 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.003959 seconds (Warm-up)
Chain 2:                0.003394 seconds (Sampling)
Chain 2:                0.007353 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'kids2' 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: 0.00386 seconds (Warm-up)
Chain 3:                0.003097 seconds (Sampling)
Chain 3:                0.006957 seconds (Total)
Chain 3:
```

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

# let's first check the estimates:
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.
```

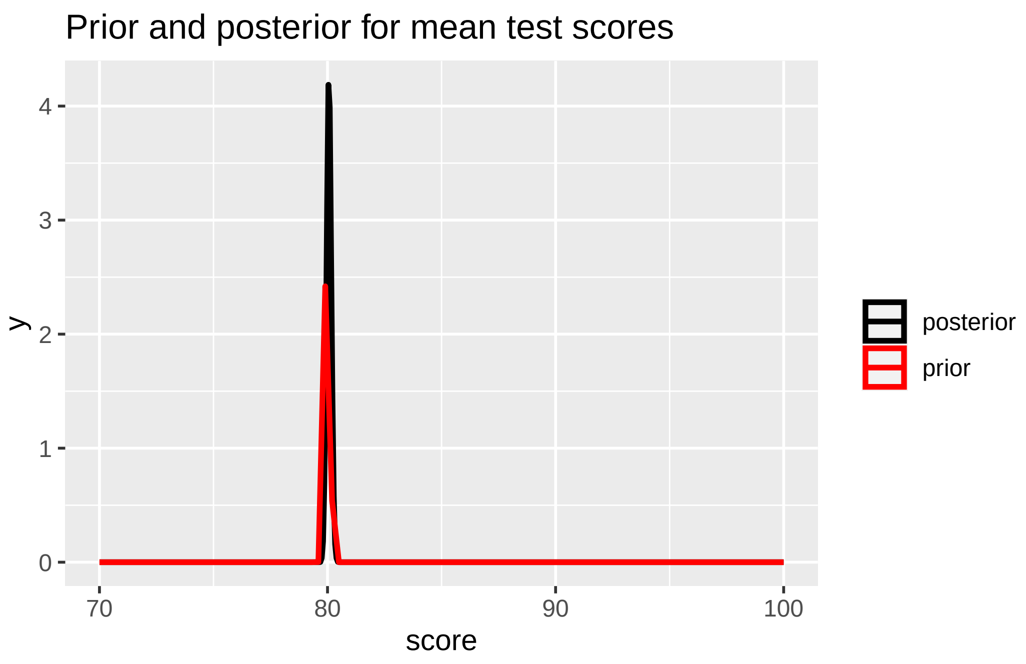
```
          mean se_mean   sd     2.5%      25%      50%      75%    97.5% n_eff
mu       80.06    0.00 0.10    79.87    80.00    80.06    80.13    80.26   683
sigma    21.47    0.03 0.76    20.06    20.93    21.42    21.98    23.01   734
lp__  -1548.39    0.06 1.04 -1551.36 -1548.75 -1548.12 -1547.65 -1547.39   352
      Rhat
mu       1
sigma    1
lp__     1

Samples were drawn using NUTS(diag_e) at Tue Feb 14 21:39:50 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 estimate changes, but doesn't change much.

Let's also look at the plot

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



as we expect, the distribution becomes much more densed with such a small standard deviation.

Adding covariates

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

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

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

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

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

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

data <- list(y = y, N = length(y),
             X =X, K = K)
fit2 <- stan(file = here("code/models/kids3.stan"),
            data = data,
            iter = 1000)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -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/4.2-arm64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/R
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'kids3' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 3.5e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.35 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: 0.085274 seconds (Warm-up)
Chain 1:                0.051369 seconds (Sampling)
Chain 1:                0.136643 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'kids3' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.1e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.11 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: 0.082311 seconds (Warm-up)
Chain 2:                0.046867 seconds (Sampling)
Chain 2:                0.129178 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'kids3' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 1.1e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.11 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: 0.079769 seconds (Warm-up)
Chain 3:                0.047552 seconds (Sampling)
Chain 3:                0.127321 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'kids3' 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: 0.084918 seconds (Warm-up)
Chain 4:                0.053805 seconds (Sampling)
Chain 4:                0.138723 seconds (Total)
Chain 4:
```

```
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      78.01    0.09 2.16    73.85    76.56    78.02    79.46    82.17
beta[1]    11.16    0.10 2.43     6.38     9.54    11.17    12.72    15.91
sigma      19.82    0.02 0.67    18.52    19.36    19.81    20.27    21.17
lp__    -1514.43    0.05 1.31 -1518.04 -1514.98 -1514.08 -1513.47 -1512.98
        n_eff Rhat
alpha     581 1.00
beta[1]   595 1.00
sigma    1065 1.00
lp__      695 1.01

Samples were drawn using NUTS(diag_e) at Tue Feb 14 21:39:59 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).
```

Question 3

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

using lm(), we have:

```r
summary(lm(kid_score ~ mom_hs, data = kidiq))
```

```

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 ***
```

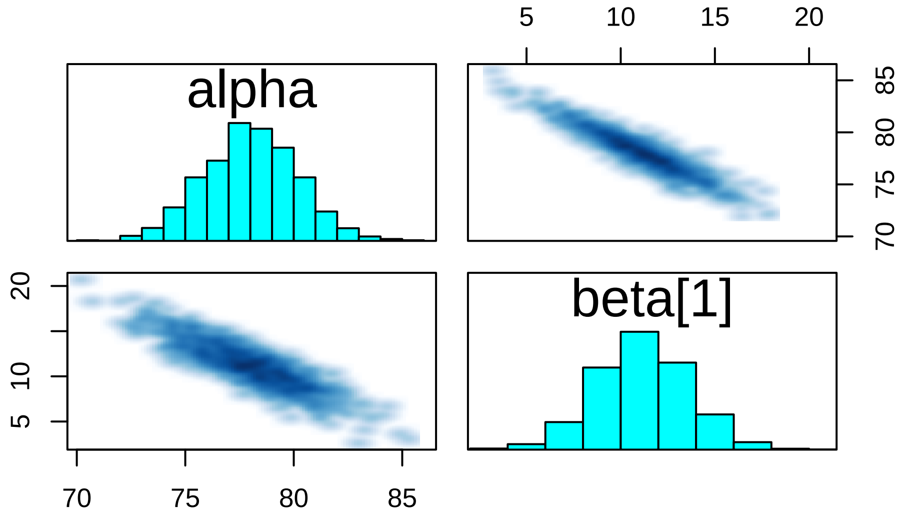
```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

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
```

which is very comparable to the above simulated results

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?

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

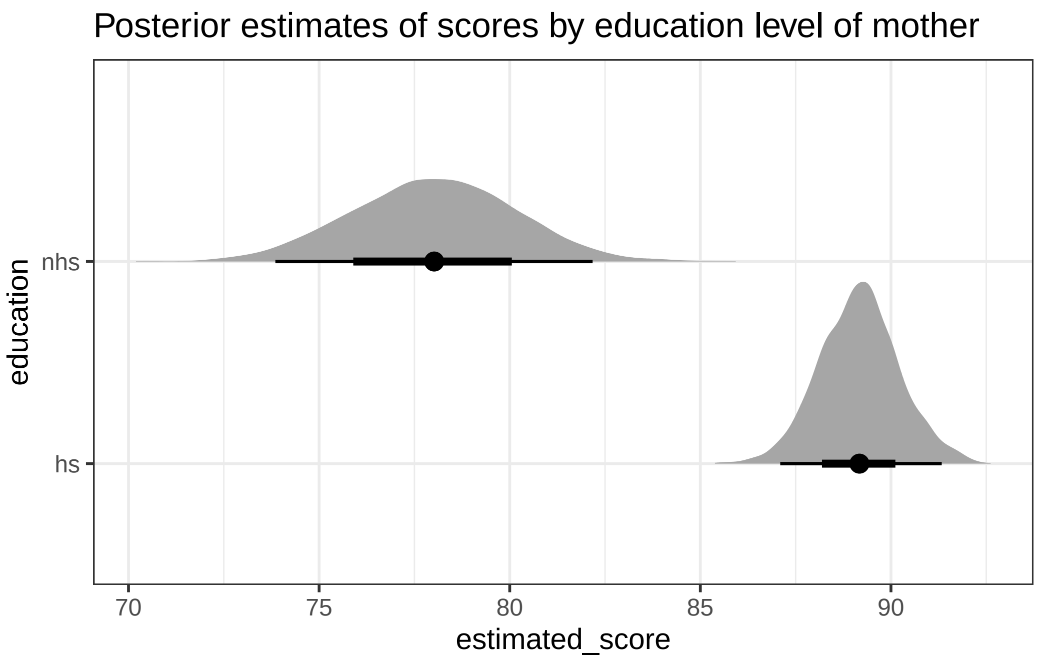


we see that although both alpha and beta (slope and intercept) are normally distributed (which by default the alpha is from Normal(0,100), and beta from Normal(0,1)), but their values are strongly correlated. This will potentially be a problem as there is collinearity problem between the two variables, which decrease the significance of the model we fitted.

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

```r
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.

let's re-run this:

```r
X1 <- as.matrix(kidiq$mom_hs, ncol = 1)
X2 <- as.matrix(kidiq$mom_iq - mean(kidiq$mom_iq), ncol = 1)
X <- cbind(X1,X2)
K <- 2
```

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

```
SAMPLING FOR MODEL 'kids3' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 2.4e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.24 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: 0.082873 seconds (Warm-up)
Chain 1:                0.0627 seconds (Sampling)
Chain 1:                0.145573 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'kids3' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.1e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.11 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: 0.111068 seconds (Warm-up)
Chain 2:                0.071254 seconds (Sampling)
Chain 2:                0.182322 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'kids3' 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: 0.109582 seconds (Warm-up)
Chain 3:                0.066025 seconds (Sampling)
Chain 3:                0.175607 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'kids3' NOW (CHAIN 4).
Chain 4:
```

```
Chain 4: Gradient evaluation took 1.1e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.11 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: 0.09084 seconds (Warm-up)
Chain 4:                0.065555 seconds (Sampling)
Chain 4:                0.156395 seconds (Total)
Chain 4:
```

and let's take a look at the summary of fit 3.

```
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      82.26    0.06 1.85    78.73    80.97    82.27    83.54    86.00
beta[1]     5.74    0.07 2.09     1.65     4.32     5.71     7.19     9.84
beta[2]     0.56    0.00 0.06     0.45     0.52     0.57     0.60     0.69
sigma      18.11    0.02 0.61    16.98    17.68    18.10    18.51    19.36
lp__    -1474.37    0.05 1.39 -1477.98 -1475.05 -1474.05 -1473.35 -1472.70
        n_eff Rhat
alpha     902    1
beta[1]   938    1
beta[2]  1521    1
sigma    1419    1
```

```
lp__      890    1

Samples were drawn using NUTS(diag_e) at Tue Feb 14 21:40:01 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 coefficent means, with other factors (mom_hs) holding the same, with one increment in the mom_iq compared to the average mom_iq, the kid_score is expected to increase by 0.56.

Question 5

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

let's look at the summary of lm()

```
df <- kidiq |>
  mutate(centred_mom_iq = mom_iq - mean(mom_iq))

summary(lm(kid_score ~ mom_hs + centred_mom_iq ,data = df))
```

```

Call:
lm(formula = kid_score ~ mom_hs + centred_mom_iq, data = df)

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 ** 
centred_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
```

as we can see, the results are very comparable.

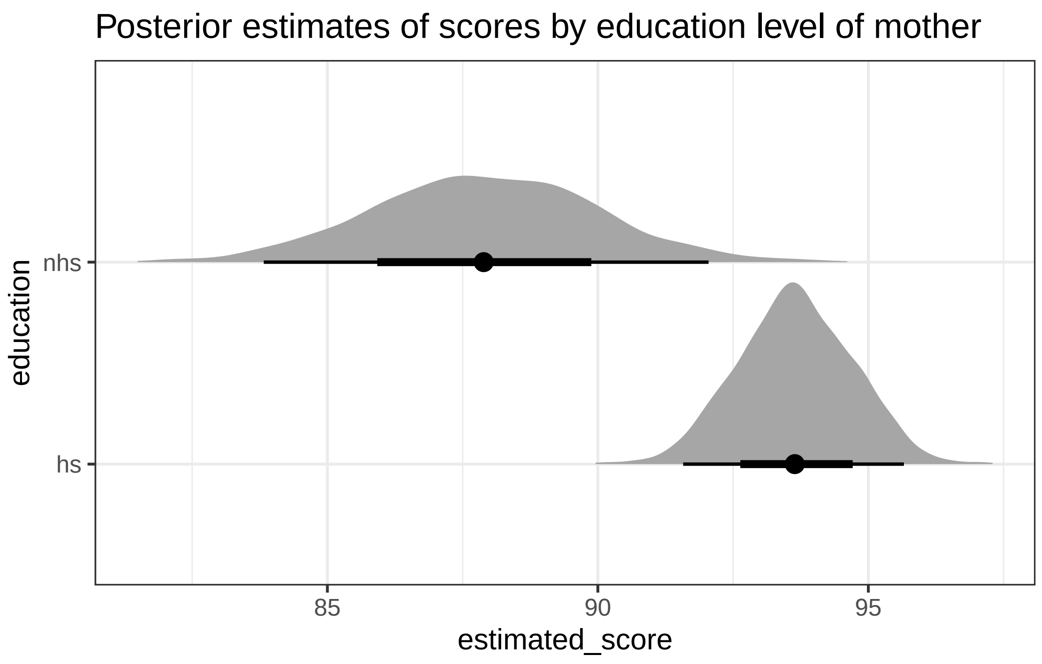
Question 6

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

Let's fit in the model of fit3

```r
miq <- mean(kidiq$mom_iq) # calculate the mean iq of moms from the beginning

fit3 |>
  spread_draws(alpha, beta[k], sigma) |>
  pivot_wider(names_from = "k", values_from = "beta", names_prefix = 'beta') |>
     mutate(nhs = alpha + beta2*(110-miq), # no high school,mom iq is 110
          hs = alpha +  beta1 + beta2*(110-miq)) |>
  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")
```

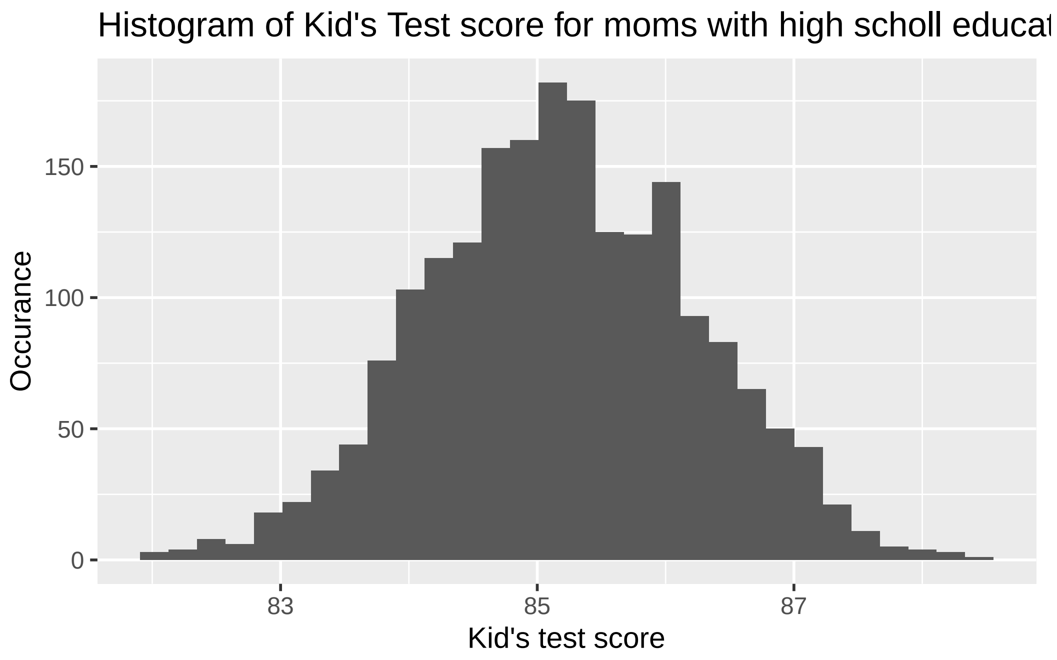


Question 7

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.

Let's plot them here:

```r
fit3 |>
  spread_draws(alpha, beta[k], sigma) |>
  pivot_wider(names_from = "k", values_from = "beta", names_prefix = 'beta') |>
  mutate(hs = alpha +  beta1 + beta2*(95-miq)) |>
  ggplot(aes(x = hs)) +
  labs(title = "Histogram of Kid's Test score for moms with high scholl education and an I
       x = "Kid's test score",
       y = "Occurance") +
  geom_histogram(bins = 30)
```


Histogram of Kid's Test score for moms with high scholl educat
Occurance
Kid's test score