lab 5

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)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(rstan)
```

```
##               : StanHeaders
```

```
## rstan (Version 2.21.3, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
```

```
##
##           : 'rstan'
```

```
##                'package:tidyr':
##
##     extract
```

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

```
## here() starts at C:/Users/1
```

```
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(psych)
```

```
##
##           : 'psych'
```

```
##                 'package:rstan':
##
##     lookup
```

```
##                  'package:ggplot2':
##
##     %+%, alpha
```

The data look like this:

```
kidiq <- read_rds("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

Figure 1

From this summary we may observe many interesting things: -se of kids score and mom's iq are relatively large, which tells us about diversity in these categories - there are 79% of mothers with completed high school in this sample - mom's average iq higher than median which mean that the distribution is right-skewed and the number of mothers with an iq higher than average is greater than a number of mothers having iq lower than average.

```
describe(kidiq)[c('vars', 'n', 'mean', 'sd', 'median', 'min', 'max', 'se')]
```

```
##           vars   n   mean    sd median   min    max   se
## kid_score    1 434  86.80 20.41  90.00 20.00 144.00 0.98
## mom_hs       2 434   0.79  0.41   1.00  0.00   1.00 0.02
## mom_iq       3 434 100.00 15.00  97.92 71.04 138.89 0.72
## mom_age      4 434  22.79  2.70  23.00 17.00  29.00 0.13
```

Figure 2

From this correlation matrix one may observe which cols have correlations (The size of circles relates to the power of correlation)

So, one may see that the strongest corr is found between mom_iq and kid_score AND between mom_iq and mom_hs.

```
corrplot(cor(kidiq), method = 'square', order = 'FPC', type = 'lower', diag = FALSE)
```

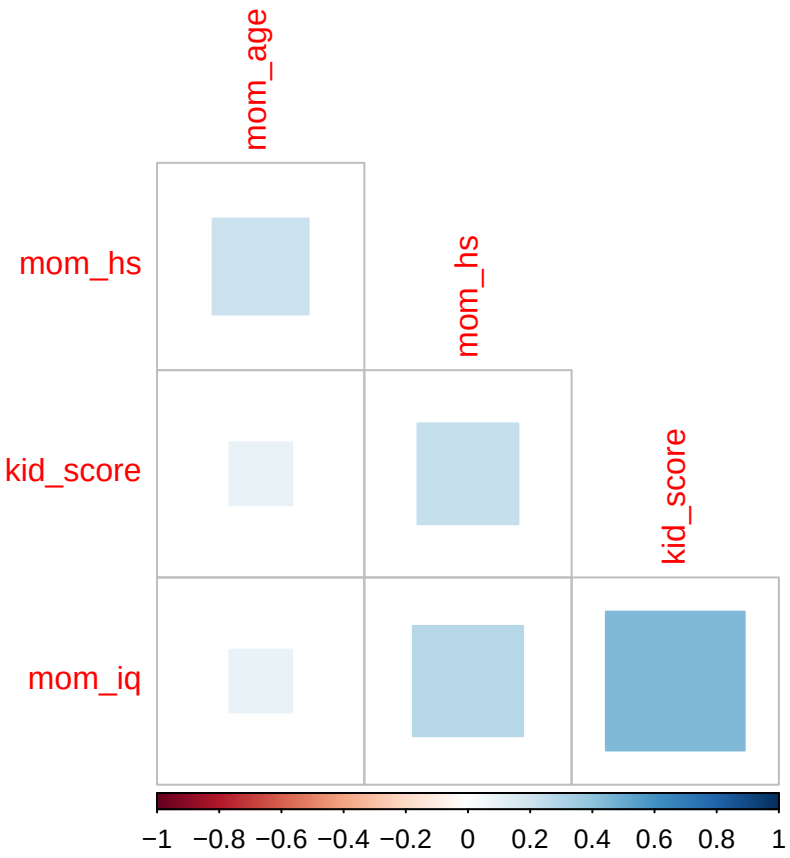
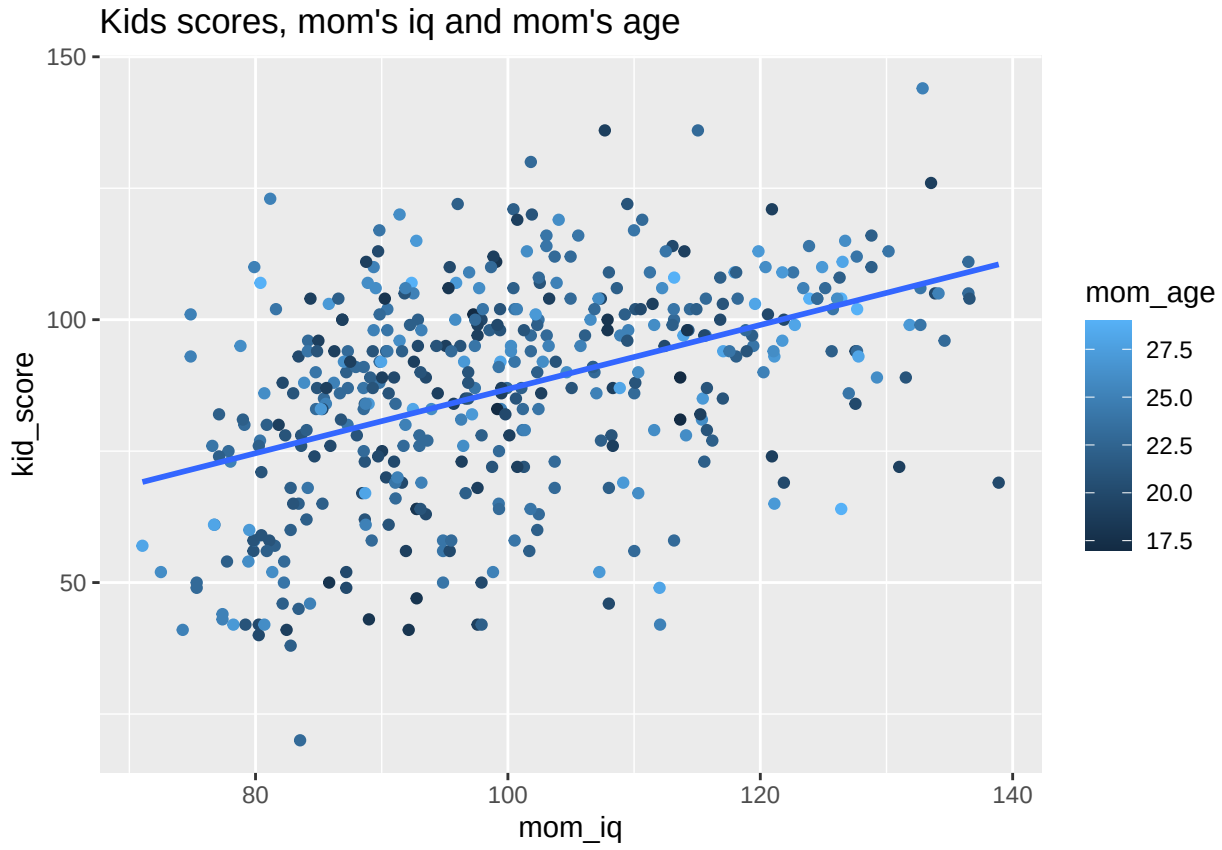


Figure 3

From this graph one may see that there is a positive correlation between mom's iq and kids performance, but the age of a mother looks random and not connected to neither mom's iq nor kids scores

```
ggplot(data = kidiq, aes(x = mom_iq, y = kid_score, color = mom_age)) +
  geom_point()+
  geom_smooth(method = 'lm', se = F)+
  ggtitle("Kids scores, mom's iq and mom's age")
```

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



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

Now we can run the model:

```
Sys.setenv(BINPREF = 'C:/rtools40/mingw64/bin/')
```

```
fit <- stan(file = "kids2.stan",
            data = data)
```

```
## Warning in readLines(file, warn = TRUE):                    'C:
## \Users\1\Documents\AS2\kids2.stan'
```

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

```
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.106 seconds (Warm-up)
## Chain 1:                0.064 seconds (Sampling)
## Chain 1:                0.17 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'kids2' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.13 seconds (Warm-up)
## Chain 2:                0.07 seconds (Sampling)
## Chain 2:                0.2 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'kids2' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
```

```
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.059 seconds (Warm-up)
## Chain 3:                0.038 seconds (Sampling)
## Chain 3:                0.097 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'kids2' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.062 seconds (Warm-up)
## Chain 4:                0.038 seconds (Sampling)
## Chain 4:                0.1 seconds (Total)
## Chain 4:
```

Look at the summary

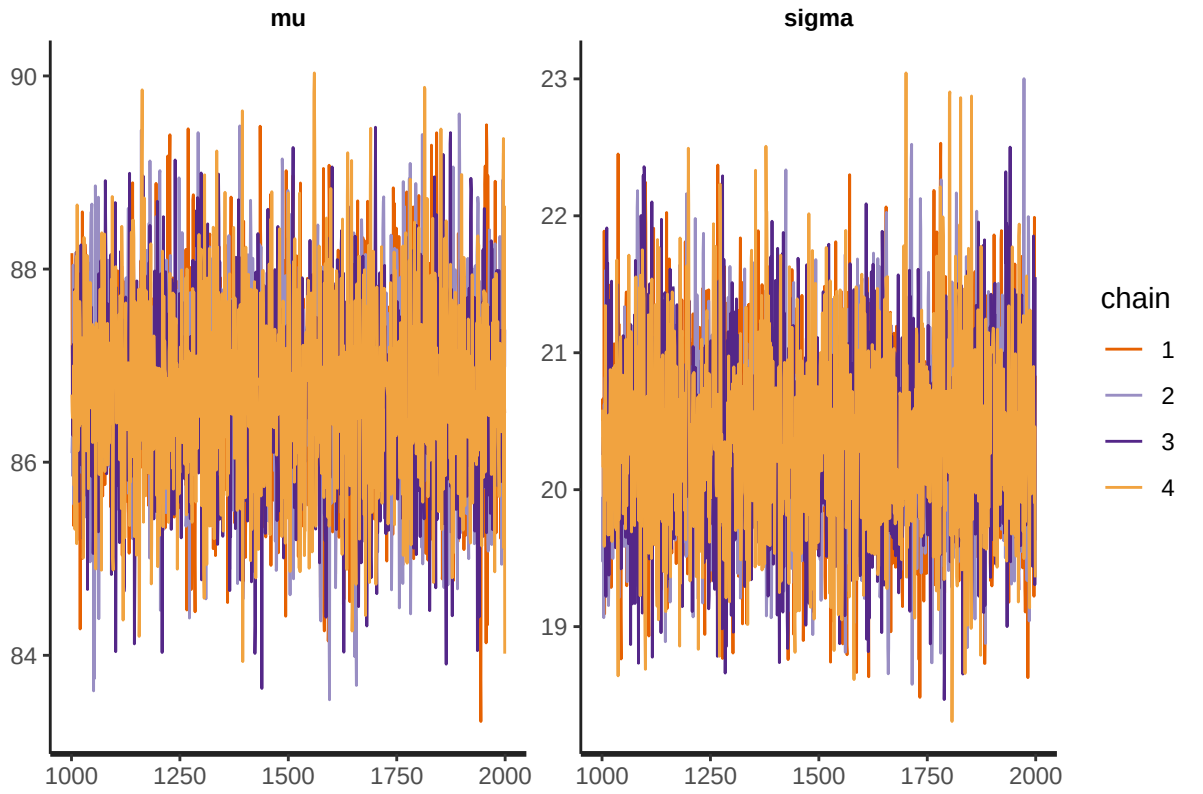
```
fit
```

```
## Inference for Stan model: kids2.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##           mean se_mean   sd     2.5%      25%      50%      75%    97.5% n_eff
## mu       86.75    0.02 0.97    84.87    86.11    86.73    87.41    88.72  3705
```

```
## sigma    20.37    0.01 0.68    19.10    19.90    20.36    20.83    21.77  3399
## lp__  -1525.74    0.02 0.99 -1528.35 -1526.11 -1525.42 -1525.04 -1524.78  1893
##       Rhat
## mu       1
## sigma    1
## lp__     1
##
## Samples were drawn using NUTS(diag_e) at Mon Feb 14 17:21:42 2022.
## 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.

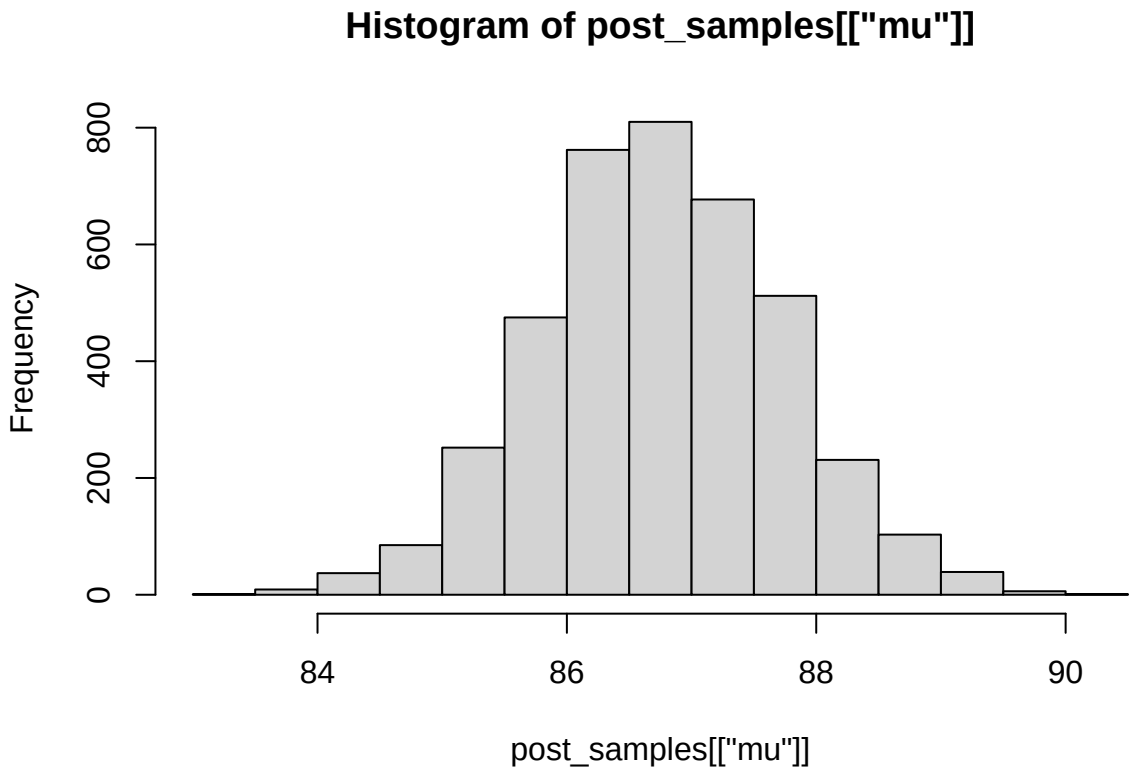
Understanding output

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 <- rstan::extract(fit)
```

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] 86.73271
```

```
quantile(post_samples[["mu"]], 0.025)
```

```
##     2.5%
## 84.87463
```

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

```
##    97.5%
## 88.71894
```

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.

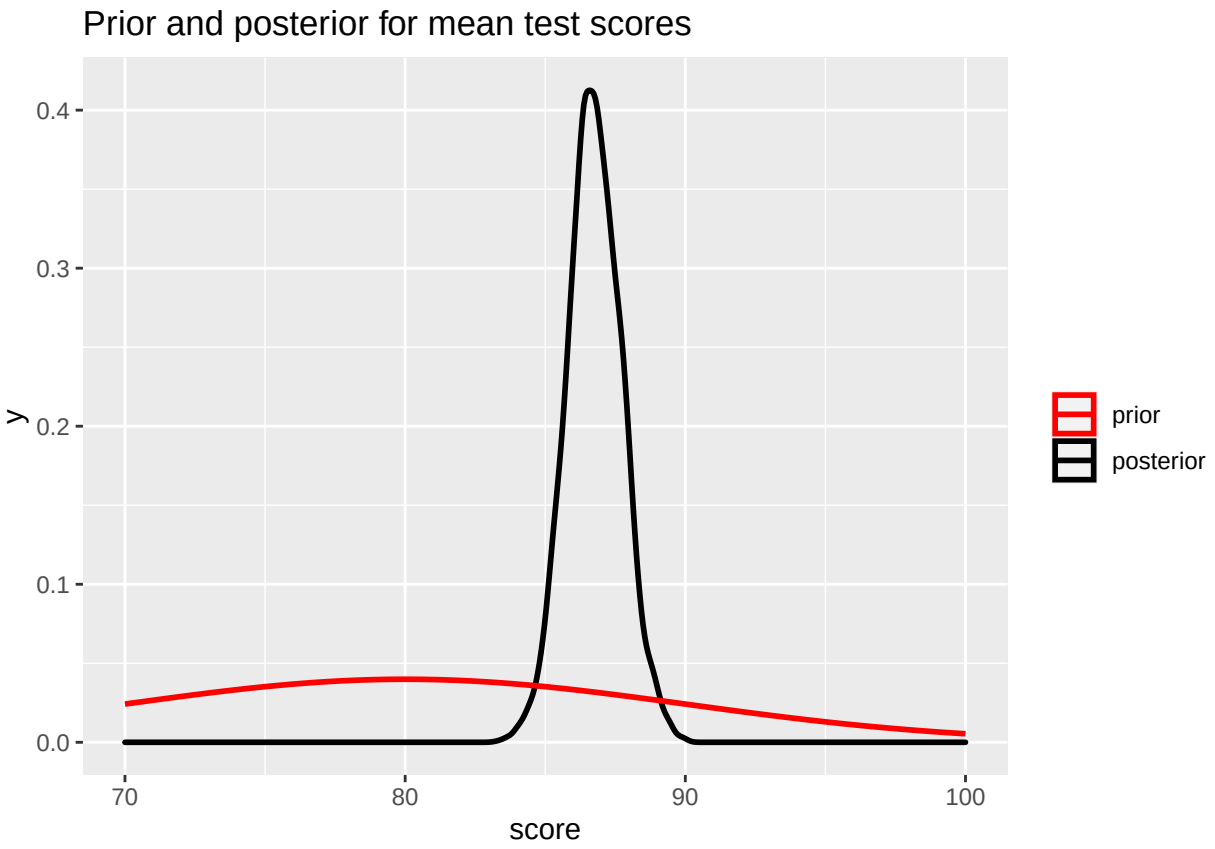
Get the posterior samples for mu and sigma in long format:

```
dsamples <- fit %>%
  gather_draws(mu, sigma)
dsamples
```

```
## # A tibble: 8,000 x 5
## # Groups:   .variable [2]
##    .chain .iteration .draw .variable .value
##     <int>      <int> <int> <chr>      <dbl>
##  1      1          1     1 mu          88.2
##  2      1          2     2 mu          86.5
##  3      1          3     3 mu          86.3
##  4      1          4     4 mu          86.8
##  5      1          5     5 mu          85.3
##  6      1          6     6 mu          87.2
##  7      1          7     7 mu          87.5
##  8      1          8     8 mu          87.6
##  9      1          9     9 mu          86.1
## 10      1         10    10 mu          86.4
## # ... with 7,990 more rows
```

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?

-Yes

Plot the prior and posterior densities.

```
y <- kidiq$kid_score
mu0 <- 80
sigma0 <- 0.1
data <- list(y = y,
             N = length(y),
             mu0 = mu0,
             sigma0 = sigma0)

fit1 <- stan(file = "kids2.stan",
            data = data)
```

```
## Warning in readLines(file, warn = TRUE):                    'C:
## \Users\1\Documents\AS2\kids2.stan'
```

```
##
## SAMPLING FOR MODEL 'kids2' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
```

```
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.051 seconds (Warm-up)
## Chain 1:                0.05 seconds (Sampling)
## Chain 1:                0.101 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'kids2' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.049 seconds (Warm-up)
## Chain 2:                0.051 seconds (Sampling)
## Chain 2:                0.1 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'kids2' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
```

```
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.05 seconds (Warm-up)
## Chain 3:                0.056 seconds (Sampling)
## Chain 3:                0.106 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'kids2' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.039 seconds (Warm-up)
## Chain 4:                0.066 seconds (Sampling)
## Chain 4:                0.105 seconds (Total)
## Chain 4:
```

```
fit1
```

```
## Inference for Stan model: kids2.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd     2.5%      25%      50%      75%    97.5% n_eff
## mu      80.06    0.00 0.10    79.87    80.00    80.07    80.13    80.26  3606
## sigma   21.43    0.01 0.73    20.06    20.93    21.40    21.91    22.91  3092
## lp__ -1548.39    0.02 1.02 -1551.17 -1548.79 -1548.07 -1547.66 -1547.39  1758
##      Rhat
## mu      1
```

```
## sigma    1
## lp__     1
##
## Samples were drawn using NUTS(diag_e) at Mon Feb 14 17:21:48 2022.
## 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(fit1)
post_samples <- rstan::extract(fit1)

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

```
## [1] 80.06536
```

```
quantile(post_samples[["mu"]], 0.025)
```

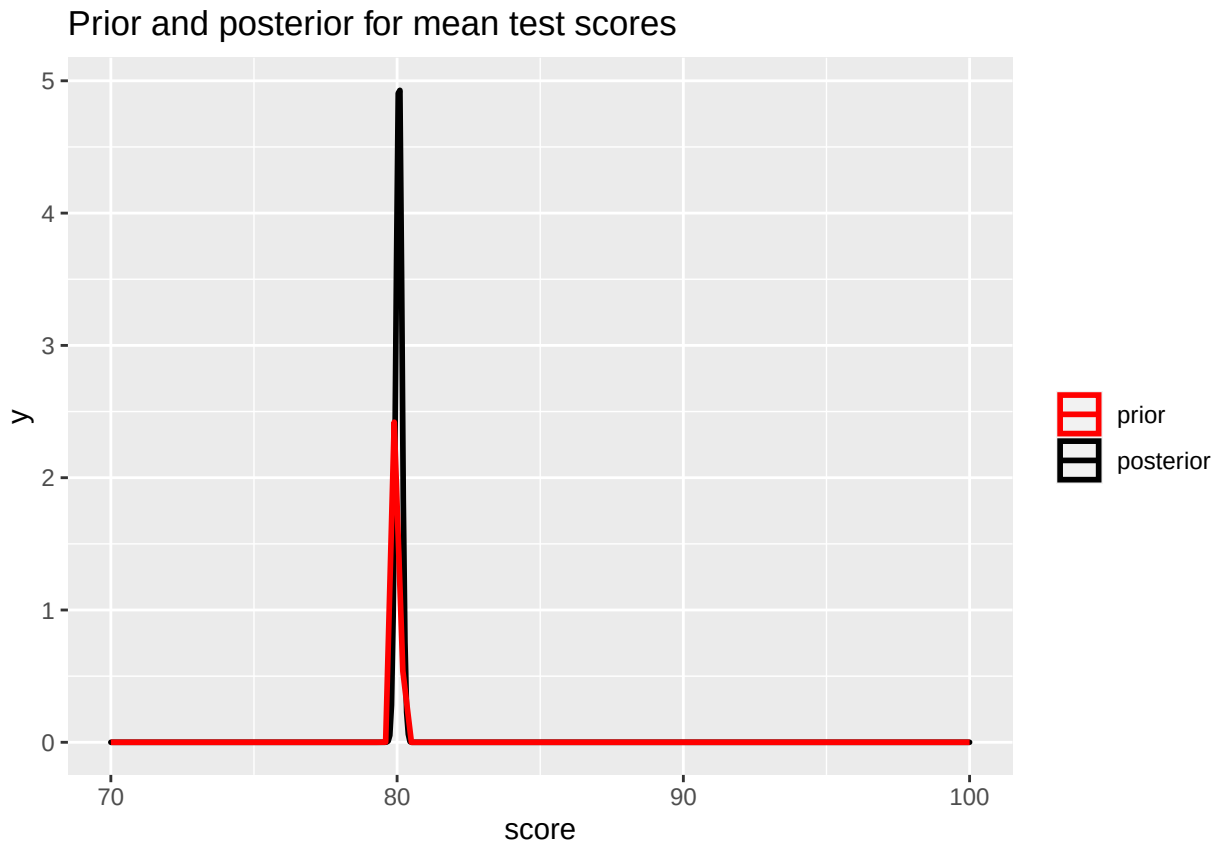
```
##     2.5%
## 79.86613
```

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

```
##    97.5%
## 80.26215
```

```
dsamples <- fit1 %>%
  gather_draws(mu, sigma)
#dsamples

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



Adding covariates

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

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

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)
K <- 1
data <- list(y = y, N = length(y),
             X =X, K = K)
fit2 <- stan(file = "kids3.stan",
            data = data,
            iter = 1000)
```

```
## Warning in readLines(file, warn = TRUE):                    'C:
## \Users\1\Documents\AS2\kids3.stan'
```

```
##
## SAMPLING FOR MODEL 'kids3' NOW (CHAIN 1).
## Chain 1:
```

```
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.561 seconds (Warm-up)
## Chain 1:                0.192 seconds (Sampling)
## Chain 1:                0.753 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'kids3' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.286 seconds (Warm-up)
## Chain 2:                0.184 seconds (Sampling)
## Chain 2:                0.47 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'kids3' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.383 seconds (Warm-up)
## Chain 3:                0.22 seconds (Sampling)
## Chain 3:                0.603 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'kids3' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.329 seconds (Warm-up)
## Chain 4:                0.202 seconds (Sampling)
## Chain 4:                0.531 seconds (Total)
## Chain 4:
```

Question 3

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

```
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      77.93    0.08 2.11    73.81    76.55    77.91    79.30    82.18
## beta[1]    11.24    0.08 2.37     6.22     9.73    11.24    12.81    15.81
## sigma      19.88    0.02 0.68    18.58    19.40    19.88    20.34    21.22
## lp__    -1514.46    0.06 1.33 -1517.95 -1515.05 -1514.09 -1513.47 -1512.97
##         n_eff Rhat
## alpha     786 1.00
## beta[1]   798 1.00
## sigma    1056 1.01
## lp__      560 1.01
##
## Samples were drawn using NUTS(diag_e) at Mon Feb 14 17:23:11 2022.
## 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).
```

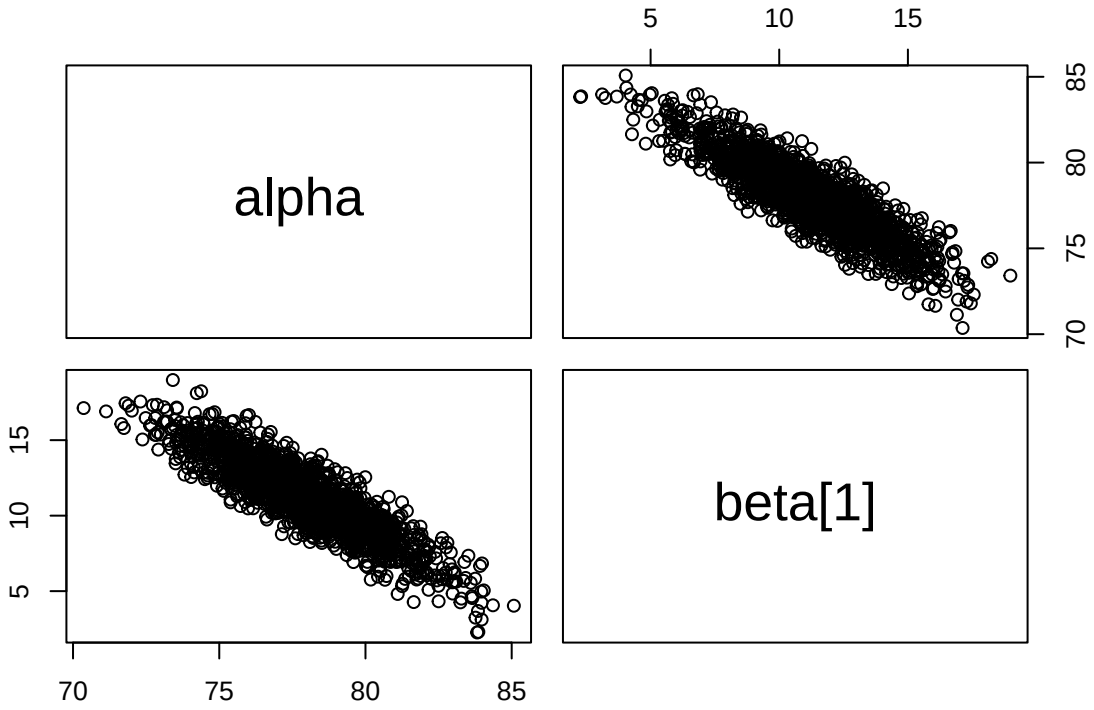
```
fit2_lm <- lm(kid_score ~ mom_hs, data = kidiq)
summary(fit2_lm)$coefficients
```

```
##             Estimate Std. Error   t value      Pr(>|t|)
## (Intercept) 77.54839   2.058612 37.670231 1.392224e-138
## mom_hs      11.77126   2.322427  5.068516  5.956524e-07
```

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?

```
dsamples2 <- fit2 %>%
  gather_draws(alpha, `beta[1]`) %>%
  #group_by(.chain, .iteration, .draw) %>%
  pivot_wider(names_from = .variable, values_from = .value) %>%
  select(alpha, `beta[1]`)
#dsamples2

pairs(dsamples2)
```



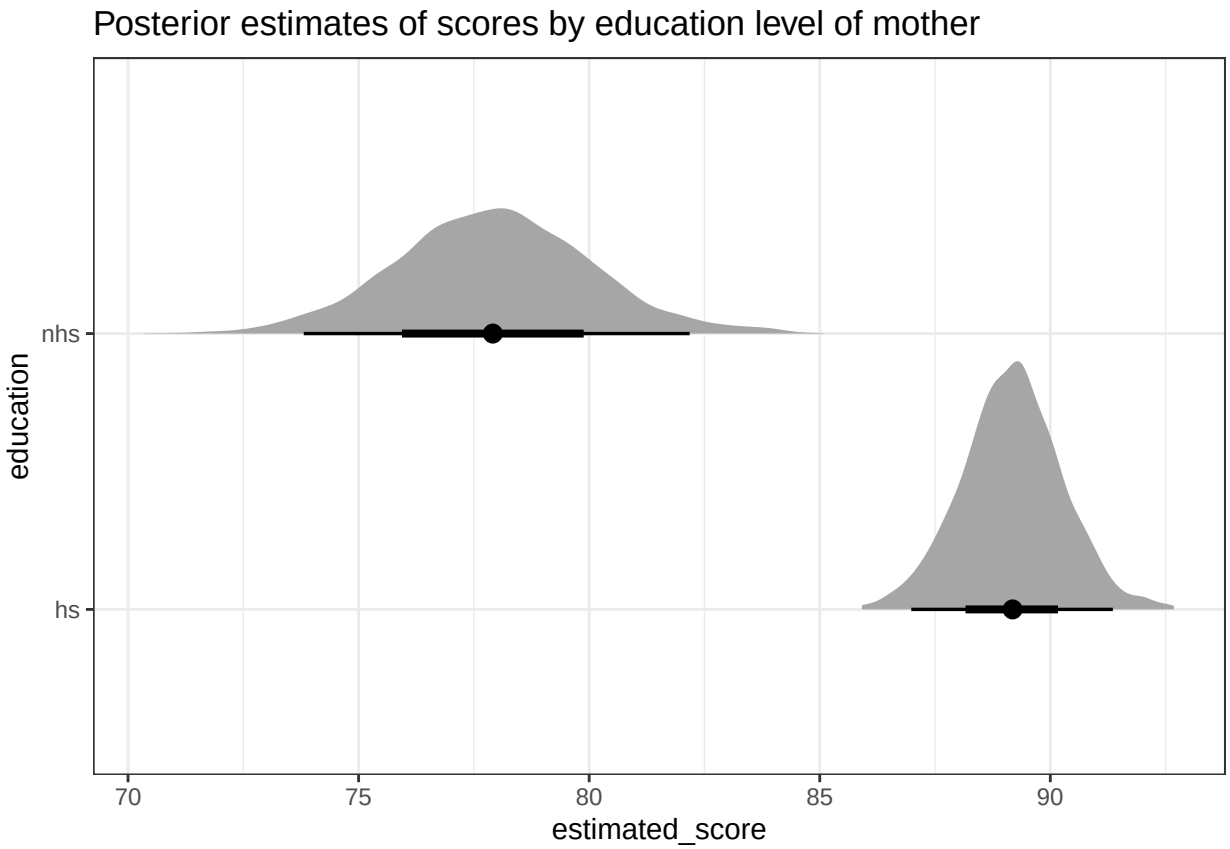
Alpha and beta are correlated. The greater the slope, the less constant. So the regression line is mooving around some points, it could be a potential problem .

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[condition], sigma) %>%
     mutate(nhs = alpha, # no high school is just the intercept
          hs = alpha + beta) %>%
  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")
```

```
## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## Please use `gather()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```



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.

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

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

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

X <- as.matrix(kidiq[c('mom_hs','mom_iq_centered')], ncol = 2)
K <- 2
data <- list(y = y, N = length(y),
             X =X, K = K)
fit3 <- stan(file = "kids3.stan",
            data = data,
            iter = 1000)
```

```
## Warning in readLines(file, warn = TRUE):                         'C:
## \Users\1\Documents\AS2\kids3.stan'
```

```
##
## SAMPLING FOR MODEL 'kids3' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.001 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 10 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: 0.34 seconds (Warm-up)
## Chain 1:                0.219 seconds (Sampling)
## Chain 1:                0.559 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'kids3' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.374 seconds (Warm-up)
## Chain 2:                0.223 seconds (Sampling)
## Chain 2:                0.597 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'kids3' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.409 seconds (Warm-up)
## Chain 3:                0.218 seconds (Sampling)
## Chain 3:                0.627 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'kids3' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.358 seconds (Warm-up)
## Chain 4:                0.241 seconds (Sampling)
## Chain 4:                0.599 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      82.28    0.06 1.92    78.50    81.00    82.34    83.56    86.09
## beta[1]     5.78    0.07 2.18     1.50     4.34     5.70     7.29    10.03
## beta[2]     0.56    0.00 0.06     0.45     0.52     0.56     0.60     0.69
## sigma      18.12    0.02 0.62    16.96    17.70    18.09    18.55    19.37
## lp__    -1474.42    0.05 1.42 -1477.95 -1475.13 -1474.06 -1473.39 -1472.67
##         n_eff Rhat
```

```
## alpha     988    1
## beta[1]   972    1
## beta[2]  1536    1
## sigma    1362    1
## lp__      830    1
##
## Samples were drawn using NUTS(diag_e) at Mon Feb 14 17:23:18 2022.
## 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).
```

If mom has iq lower that average, the scores of a kid would be lower on the (iq_diff)*0.57 in comparison with the kid's scores of mothers with average iq.

If mom has iq higher that average, the scores of a kid would be greater on the (iq_diff)*0.57 in comparison with the kid's scores of mothers with average iq.

One point increasing of mom's iq will lead to increase of kid's scores on 0.57 points.

Question 5

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

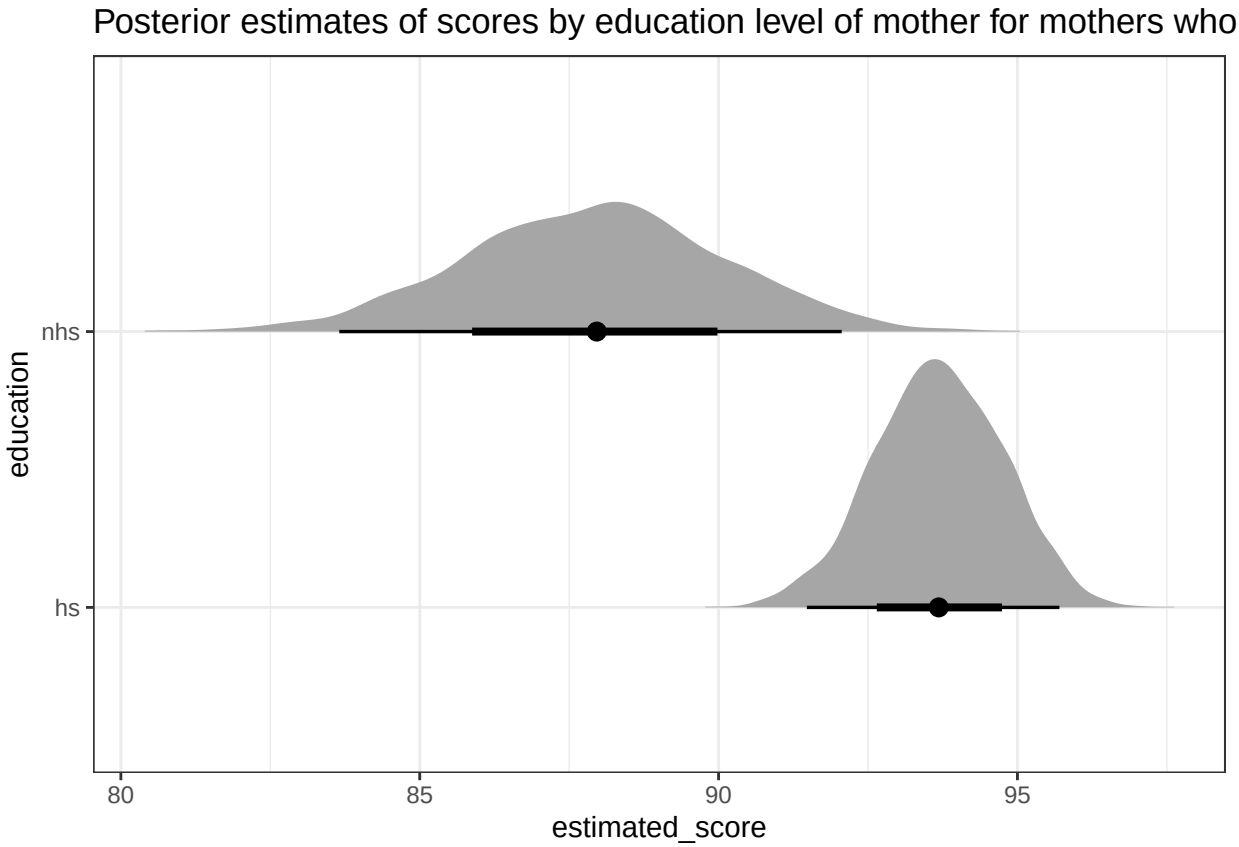
```
fit3_lm <- lm(kid_score ~ mom_hs + mom_iq_centered, data = kidiq)
summary(fit3_lm)$coefficients
```

```
##                 Estimate Std. Error   t value      Pr(>|t|)
## (Intercept)     82.122143 1.94370047 42.250411 2.435765e-155
## mom_hs           5.950117 2.21181218  2.690155  7.419327e-03
## mom_iq_centered  0.563906 0.06057408  9.309362  6.609618e-19
```

Question 6

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

```
fit3 %>%
  gather_draws(alpha, `beta[1]`, `beta[2]`, sigma) %>%
  pivot_wider(names_from = .variable, values_from = .value) %>%
  #select(alpha, `beta[1]`, `beta[2]`, sigma) %>%
  rename(beta_hs = `beta[1]`,
         beta_mom_iq = `beta[2]`) %>%
     mutate(nhs = alpha + 0*beta_hs + 10*beta_mom_iq, # no high school is just the intercept
          hs = alpha + 1*beta_hs + 10*beta_mom_iq) %>%
  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 for mothers who have an IQ of 110
```

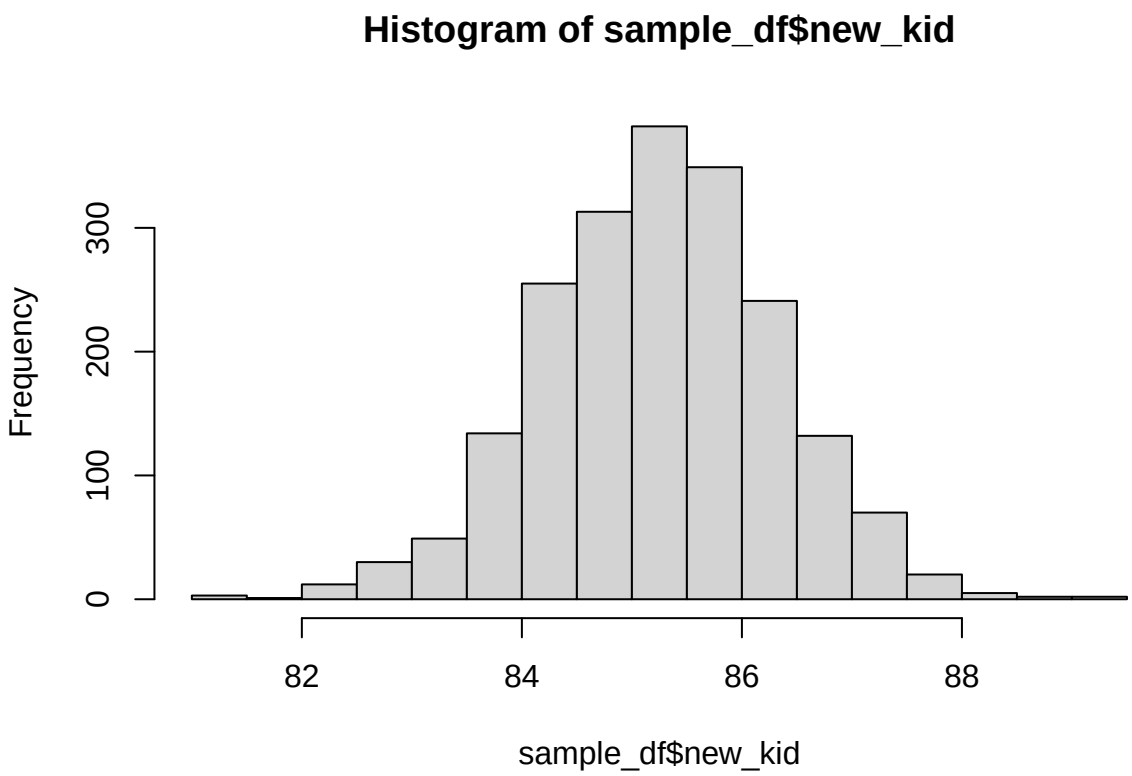


Question 7

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

```
sample_df <- fit3 %>%
  gather_draws(alpha, `beta[1]`, `beta[2]`, sigma) %>%
  pivot_wider(names_from = .variable, values_from = .value) %>%
  #select(alpha, `beta[1]`, `beta[2]`, sigma) %>%
  rename(beta_mom_hs = `beta[1]`,
         beta_mom_iq = `beta[2]`) %>%
     mutate(new_kid = alpha + beta_mom_hs + (-5)*beta_mom_iq)

hist(sample_df$new_kid)
```


Histogram of sample_df$new_kid
Frequency
0
100
200
300
82
84
86
88
sample_df$new_kid