Week 5: Bayesian linear regression and introduction to Stan

13/02/23

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)
library(ggplot2)
```

The data look like this:

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

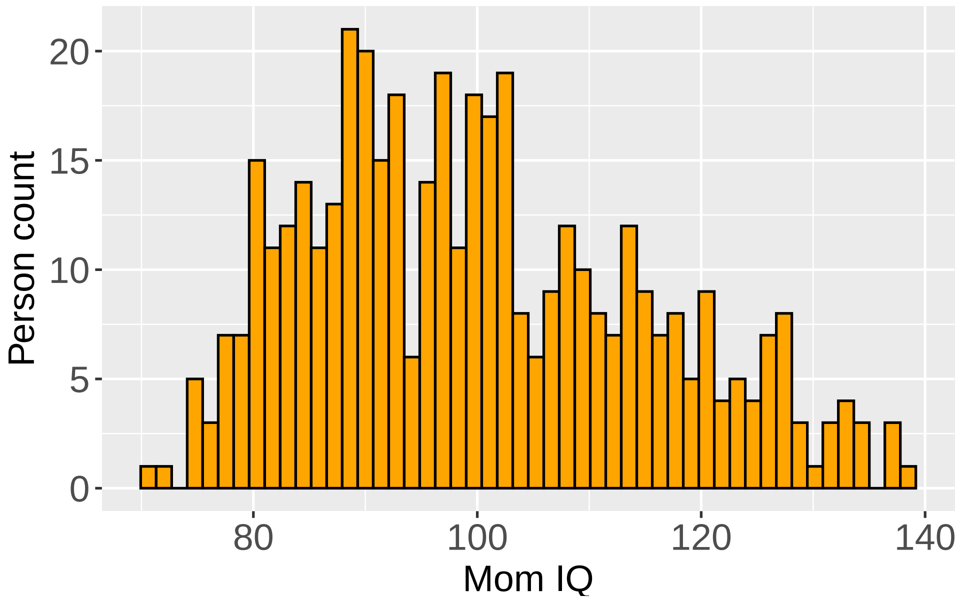
Question 1

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

```
theme_large_text <- theme(axis.text=element_text(size=15), axis.title=element_text(size=15
```

```r
kidiq %>% ggplot(aes(x=mom_iq)) +
  geom_histogram(bins=50, color="black", fill="orange") +
  labs(x='Mom IQ', y='Person count') +
  theme_large_text
```

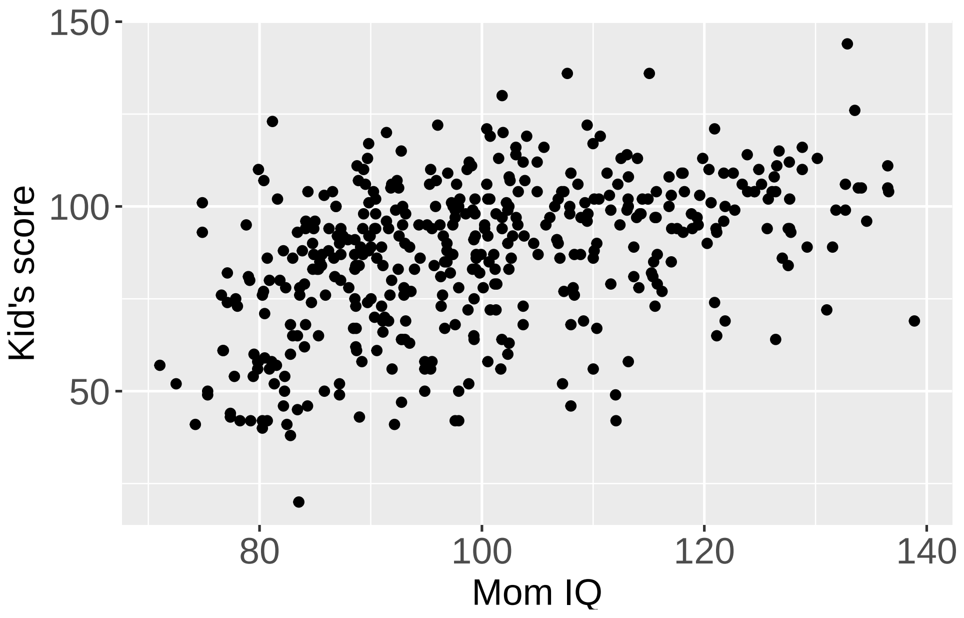


This figure shows that the mother's IQ has an asymmetric distribution with a median close to 100.

```r
median(kidiq$mom_iq)
```

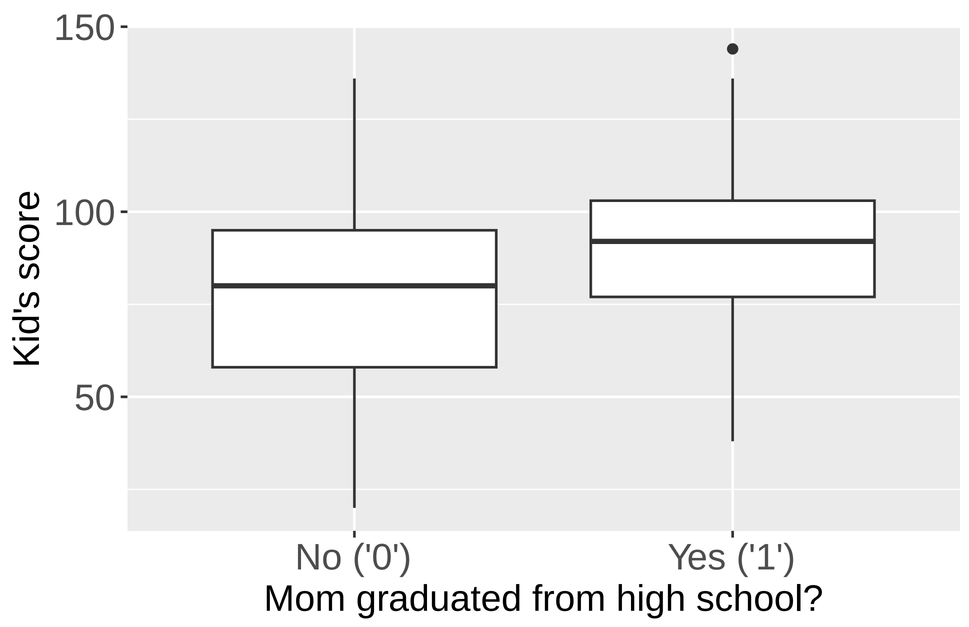
```
[1] 97.91525
```

```r
kidiq %>% ggplot(aes(x=mom_iq, y=kid_score)) +
  geom_point(stroke=0.5) +
  labs(x='Mom IQ', y="Kid's score") +
  theme_large_text
```



This figure shows that the mother's IQ has an almost linear relationship with the kid's score.

```
kidiq %>% ggplot(aes(x=as.factor(mom_hs), y=kid_score)) +
  geom_boxplot() +
  labs(x='Mom graduated from high school?', y="Kid's score") +
  scale_x_discrete(labels=c("0" = "No ('0')", "1" = "Yes ('1')")) +
  theme_large_text
```



This figures shows that the mother's education level has some positive correlation with the kid's score.

Estimating mean, no covariates

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

```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("./Lab5/kids2.stan"),
            data = data,
            chains = 3,
            iter = 500)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 3.4e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.34 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: 0.013 seconds (Warm-up)
Chain 1:                0.005 seconds (Sampling)
Chain 1:                0.018 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 7e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.07 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: 0.015 seconds (Warm-up)
Chain 2:                0.006 seconds (Sampling)
Chain 2:                0.021 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 6e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.06 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: 0.014 seconds (Warm-up)
Chain 3:                0.006 seconds (Sampling)
Chain 3:                0.02 seconds (Total)
Chain 3:
```

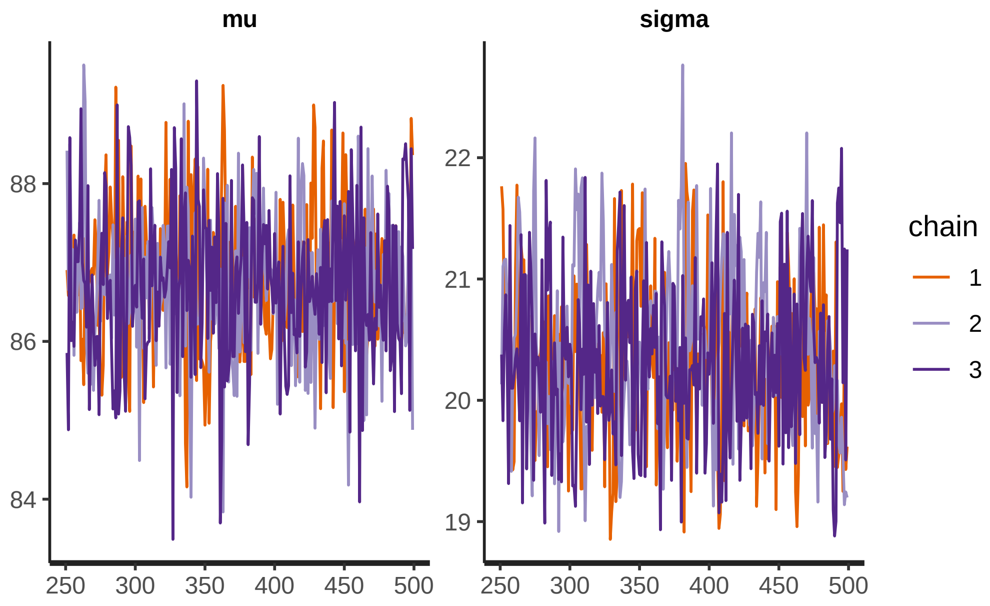
Look at the summary

```
fit
```

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

          mean se_mean   sd     2.5%      25%      50%      75%    97.5% n_eff
mu       86.74    0.03 0.94    85.02    86.09    86.71    87.34    88.65   726
sigma    20.36    0.03 0.68    19.14    19.87    20.35    20.81    21.75   425
lp__  -1525.72    0.05 0.95 -1528.17 -1526.17 -1525.40 -1525.01 -1524.77   387
      Rhat
mu       1
sigma    1
lp__     1

Samples were drawn using NUTS(diag_e) at Mon Feb 13 14:49:50 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).
```

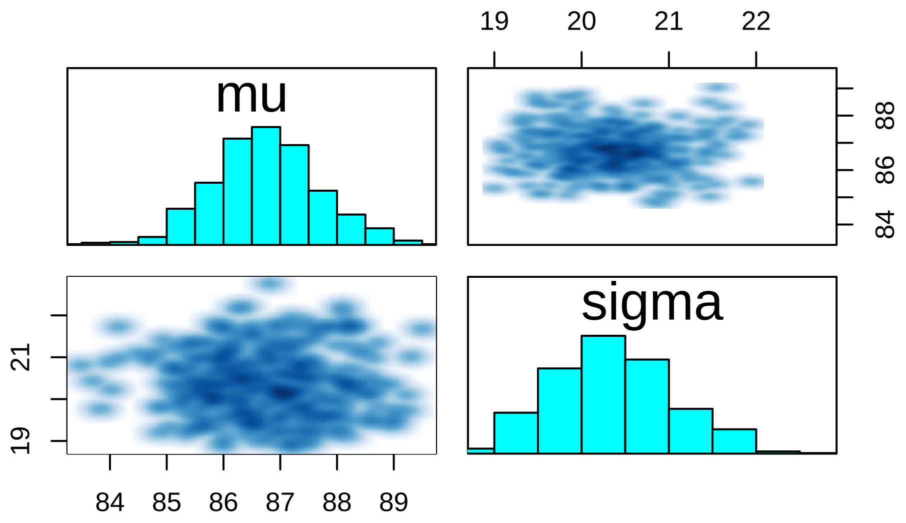
Traceplot

```
traceplot(fit)
```

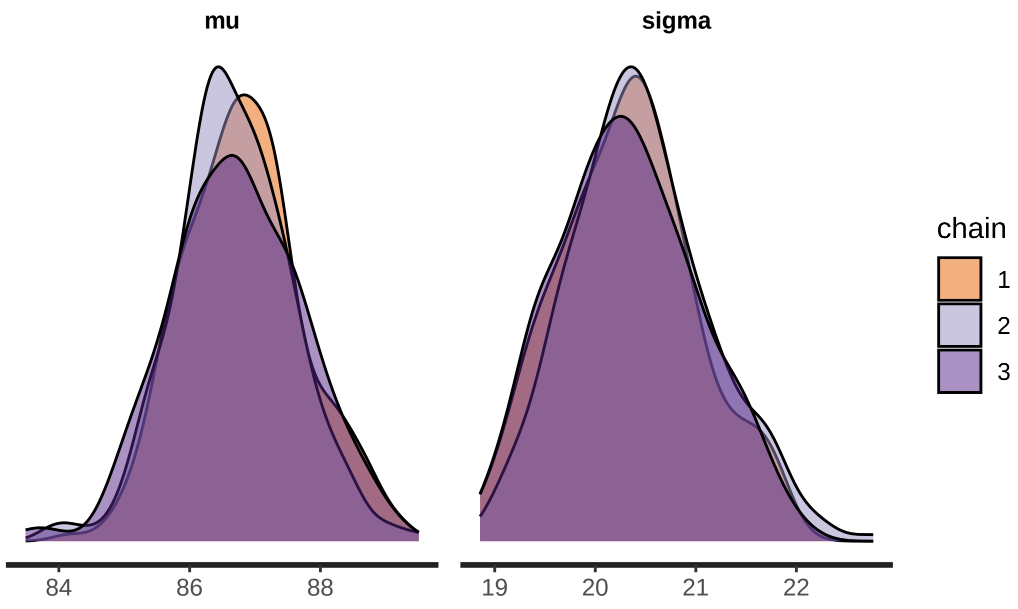


All looks fine.

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



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



Understanding output

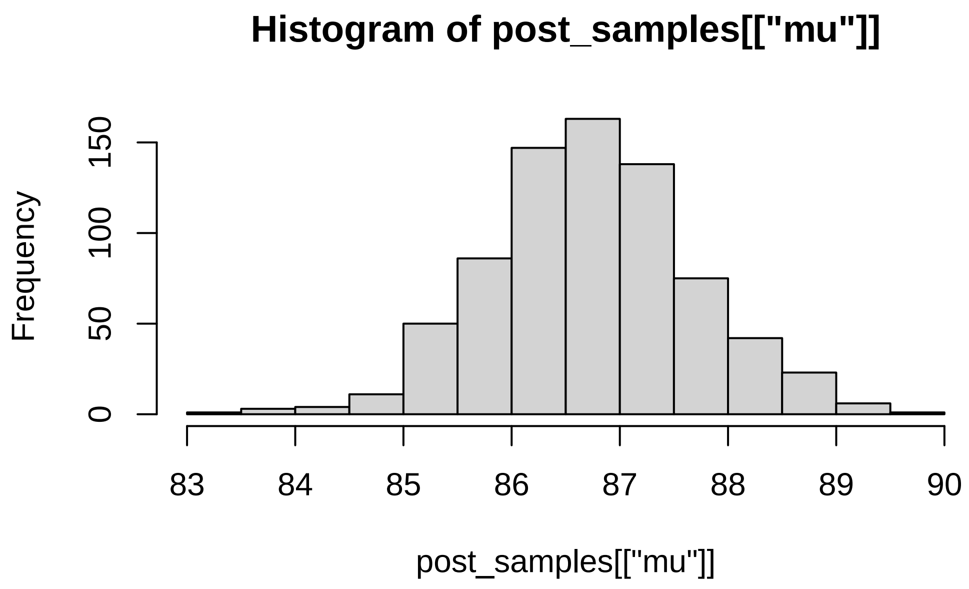
What does the model actually give us? A number of samples from the posteriors. To see this, we can use `extract` to get the samples.

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

```
[1] 86.61550 86.56237 88.05504 85.42235 87.12139 88.64193
```

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"]])
```



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

```
[1] 86.71205
```

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

```
    2.5%
85.02045
```

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

```
   97.5%
88.65223
```

Plot estimates

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

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

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

```
# A tibble: 1,500 x 5
# Groups:   .variable [2]
   .chain .iteration .draw .variable .value
    <int>      <int> <int> <chr>      <dbl>
 1      1          1     1 mu          86.9
 2      1          2     2 mu          86.6
 3      1          3     3 mu          87.3
 4      1          4     4 mu          86.8
 5      1          5     5 mu          86.8
 6      1          6     6 mu          87.3
 7      1          7     7 mu          86.9
 8      1          8     8 mu          86.9
 9      1          9     9 mu          86.6
10      1         10    10 mu          87.3
# ... with 1,490 more rows
# i Use `print(n = ...)` to see more rows
```

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

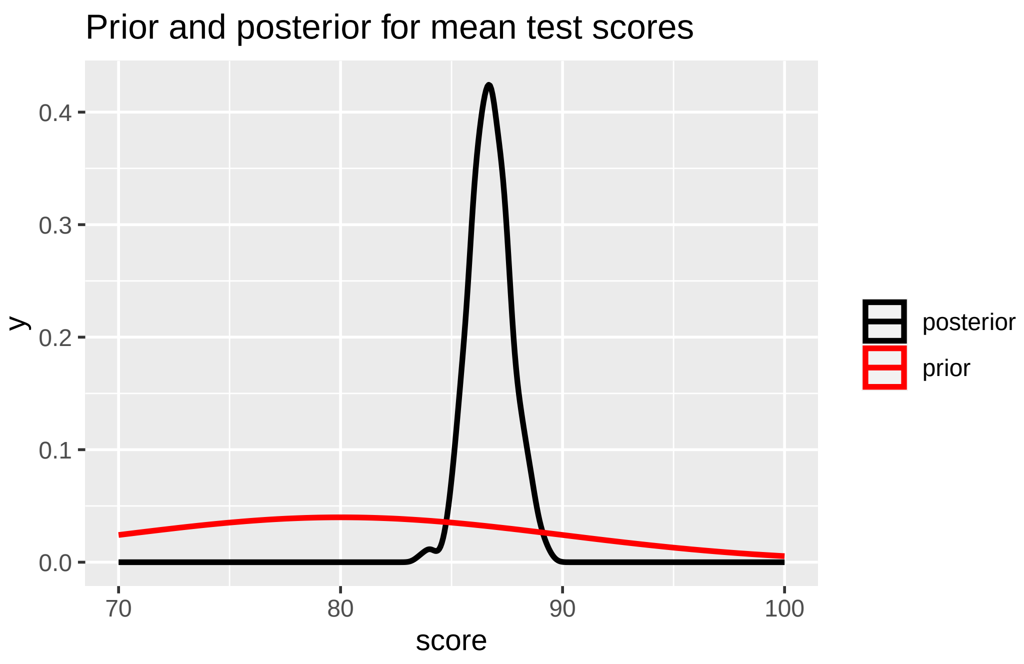
```
# A tibble: 750 x 5
   .chain .iteration .draw    mu sigma
    <int>      <int> <int> <dbl> <dbl>
 1      1          1     1  86.9  21.8
 2      1          2     2  86.6  21.6
 3      1          3     3  87.3  20.6
 4      1          4     4  86.8  20.8
 5      1          5     5  86.8  20.8
 6      1          6     6  87.3  20.8
 7      1          7     7  86.9  20.8
 8      1          8     8  86.9  20.2
 9      1          9     9  86.6  19.4
10      1         10    10  87.3  19.5
# ... with 740 more rows
# i Use `print(n = ...)` to see more rows
```

```r
# quickly calculate the quantiles using

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

```
# A tibble: 2 x 7
  .variable .value .lower .upper .width .point .interval
  <chr>      <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
1 mu          86.7   85.6   88.0    0.8 median qi
2 sigma       20.3   19.5   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.

```r
mu0 <- 80
sigma0 <- 0.1

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

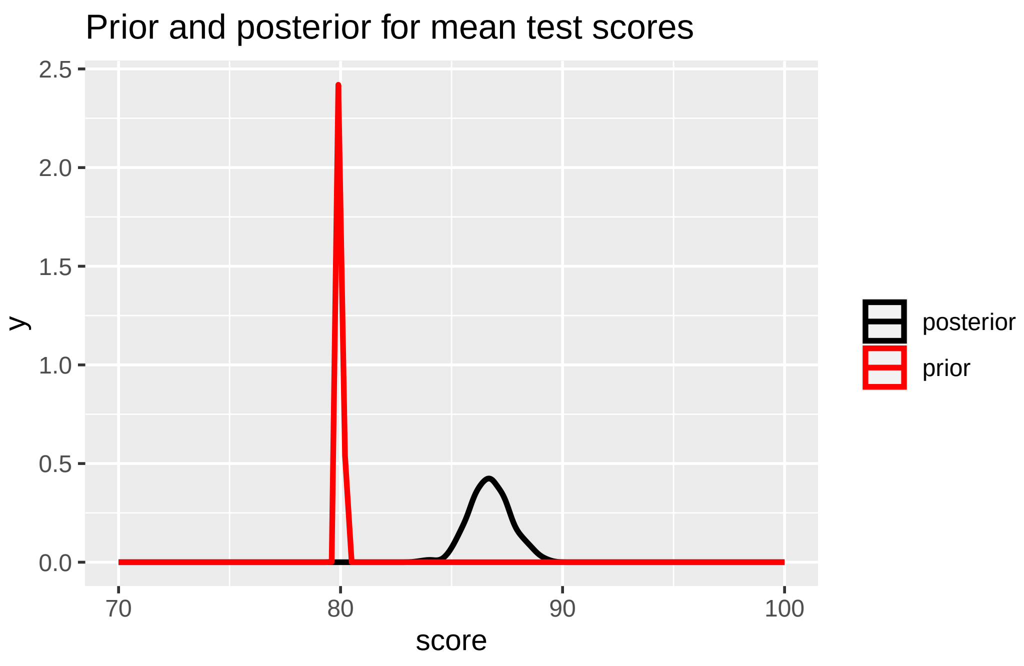
```r
fit <- stan(file = here("./Lab5/kids2.stan"),
            data = data,
            chains = 3,
            iter = 500)
```

```
SAMPLING FOR MODEL 'anon_model' 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: 0.011 seconds (Warm-up)
Chain 1:                0.006 seconds (Sampling)
Chain 1:                0.017 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 7e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 2: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 2: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 2: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 2: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 2: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 2: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 2: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 2: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 2: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 2: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 2: Iteration: 500 / 500 [100%]  (Sampling)
Chain 2:
```

```
Chain 2:  Elapsed Time: 0.008 seconds (Warm-up)
Chain 2:                0.007 seconds (Sampling)
Chain 2:                0.015 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 7e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.07 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: 0.008 seconds (Warm-up)
Chain 3:                0.007 seconds (Sampling)
Chain 3:                0.015 seconds (Total)
Chain 3:
```

```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")
```



```r
fit
```

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

          mean se_mean   sd     2.5%      25%      50%      75%    97.5% n_eff
mu       80.06    0.00 0.11    79.85    79.99    80.06    80.13    80.27   718
sigma    21.40    0.03 0.70    20.00    20.90    21.41    21.87    22.75   743
lp__  -1548.42    0.06 1.02 -1551.03 -1548.89 -1548.12 -1547.65 -1547.39   307
      Rhat
mu       1
sigma    1
lp__     1

Samples were drawn using NUTS(diag_e) at Mon Feb 13 14:50:04 2023.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at 
convergence, Rhat=1).
```

The estimates are now different from before (mu changed from 86.63 to 80.06, sigma changed from 20.25 to 21.42).

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("./Lab5/kids3.stan"),
            data = data,
            iter = 1000)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 9.4e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.94 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: 0.228 seconds (Warm-up)
Chain 1:                0.104 seconds (Sampling)
Chain 1:                0.332 seconds (Total)
Chain 1: 

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2: 
Chain 2: Gradient evaluation took 2.2e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.22 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: 0.256 seconds (Warm-up)
Chain 2:                0.139 seconds (Sampling)
Chain 2:                0.395 seconds (Total)
Chain 2: 

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3: 
Chain 3: Gradient evaluation took 2.1e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.21 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: 0.184 seconds (Warm-up)
Chain 3:                0.1 seconds (Sampling)
Chain 3:                0.284 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 3.2e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.32 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: 0.198 seconds (Warm-up)
Chain 4:                0.113 seconds (Sampling)
Chain 4:                0.311 seconds (Total)
Chain 4:
```

```
fit2
```

```
Inference for Stan model: anon_model.
```

```
4 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=2000.

            mean se_mean   sd     2.5%      25%      50%      75%    97.5%
alpha      77.98    0.07 2.00    74.10    76.61    77.98    79.35    81.84
beta[1]    11.22    0.08 2.29     6.73     9.65    11.25    12.80    15.52
sigma      19.82    0.02 0.68    18.57    19.37    19.80    20.25    21.23
lp__    -1514.40    0.05 1.31 -1517.89 -1514.94 -1514.05 -1513.48 -1512.96
        n_eff Rhat
alpha     855 1.00
beta[1]   846 1.00
sigma    1157 1.01
lp__      724 1.00

Samples were drawn using NUTS(diag_e) at Mon Feb 13 14:51:21 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()`

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

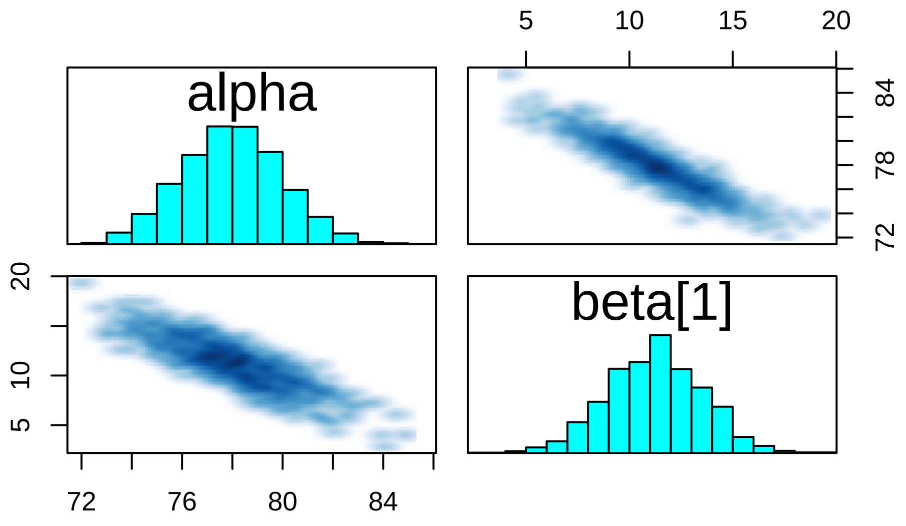
```
Call:
lm(formula = kid_score ~ mom_hs, data = kidiq)

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
```

The intercept from the `lm()` is ~ 77.54 similar to the 77.95 from bayesian regression.

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"))
```

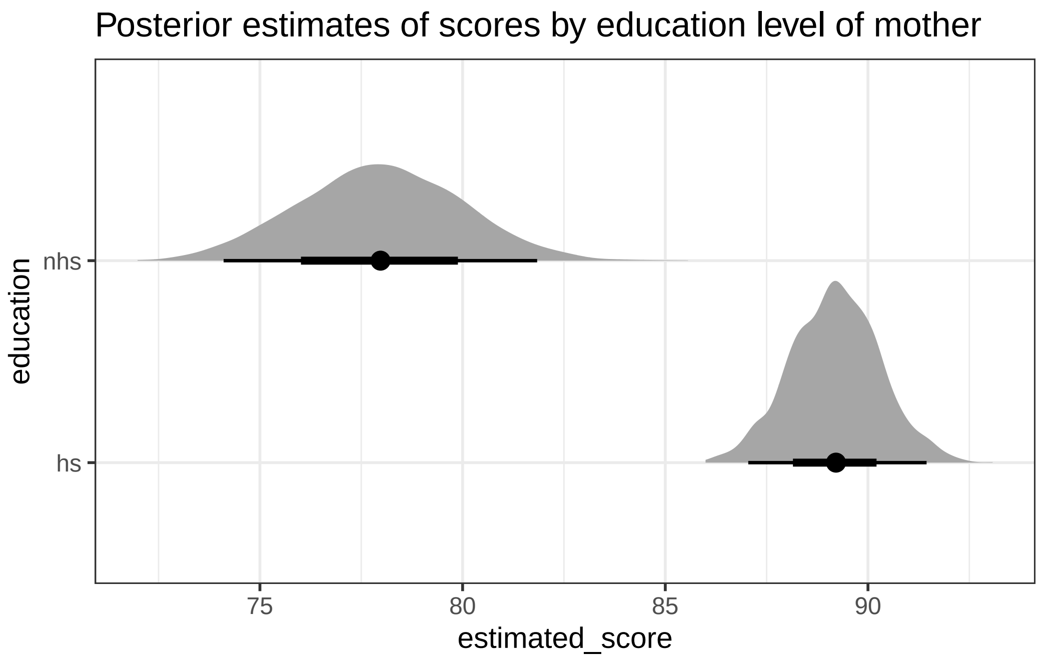


Alpha and beta are strongly anticorrelated and it's a problem.

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[k], sigma) |>
     mutate(nhs = alpha, # no high school is just the intercept
          hs = alpha + beta) |>
  dplyr::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.

```
kidiq$mom_iq_mc = with(kidiq, mom_iq - mean(kidiq$mom_iq))

X <- as.matrix(dplyr::select(kidiq, mom_hs, mom_iq_mc), ncol = 2) # force this to be a mat
K <- 2

data <- list(y = y, N = length(y),
             X =X, K = K)
fit3 <- stan(file = here("./Lab5/kids3.stan"),
            data = data,
            iter = 1000)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 3e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.3 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: 0.302 seconds (Warm-up)
Chain 1:                0.164 seconds (Sampling)
Chain 1:                0.466 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 2.2e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.22 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: 0.22 seconds (Warm-up)
Chain 2:                0.148 seconds (Sampling)
Chain 2:                0.368 seconds (Total)
Chain 2: 

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3: 
Chain 3: Gradient evaluation took 2.2e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.22 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: 0.222 seconds (Warm-up)
Chain 3:                0.135 seconds (Sampling)
Chain 3:                0.357 seconds (Total)
Chain 3: 

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4: 
Chain 4: Gradient evaluation took 2.3e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.23 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: 0.253 seconds (Warm-up)
Chain 4:                0.091 seconds (Sampling)
Chain 4:                0.344 seconds (Total)
Chain 4: 
```

```
fit3
```

```
Inference for Stan model: anon_model.
4 chains, each with iter=1000; warmup=500; thin=1; 
post-warmup draws per chain=500, total post-warmup draws=2000.

            mean se_mean   sd     2.5%      25%      50%      75%    97.5%
alpha      82.23    0.06 1.94    78.51    80.93    82.25    83.51    86.10
beta[1]     5.78    0.07 2.20     1.60     4.28     5.77     7.28    10.06
beta[2]     0.57    0.00 0.06     0.45     0.53     0.57     0.61     0.68
sigma      18.14    0.02 0.64    16.99    17.68    18.12    18.55    19.53
lp__    -1474.51    0.06 1.49 -1478.51 -1475.22 -1474.16 -1473.41 -1472.70
        n_eff Rhat
alpha    1010 1.00
beta[1]   943 1.00
beta[2]  1431 1.00
sigma    1289 1.00
lp__      586 1.01

Samples were drawn using NUTS(diag_e) at Mon Feb 13 14:51:30 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 5

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

```r
kid_mom_hsiq <- lm(kid_score ~ mom_hs + mom_iq_mc, data = kidiq)
summary(kid_mom_hsiq)
```

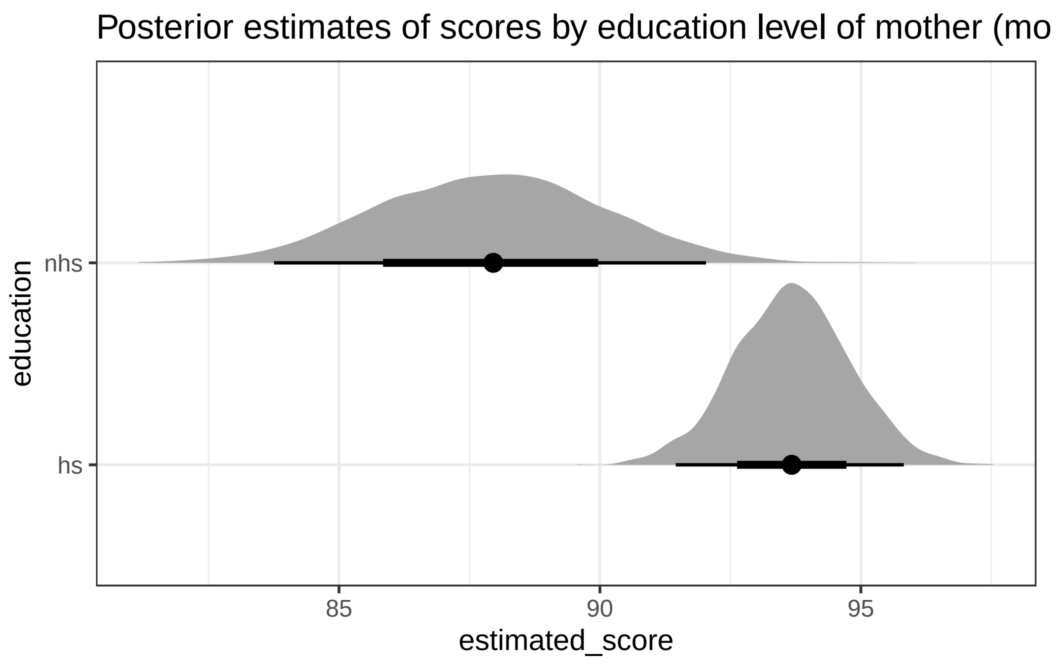
```

Call:
lm(formula = kid_score ~ mom_hs + mom_iq_mc, 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 ***
mom_hs       5.95012    2.21181   2.690  0.00742 ** 
mom_iq_mc    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
```

The intercept from `lm()` is ~ 82.12, while that from the Bayesian regression model is 82.31. They agree well.

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) |>
     mutate(nhs = alpha + `2`*(110 - mean(kidiq$mom_iq)),
            hs = alpha + `1` + `2`*(110 - mean(kidiq$mom_iq))) |> # high iq of 110
  dplyr::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 (mom IQ=110)")
```



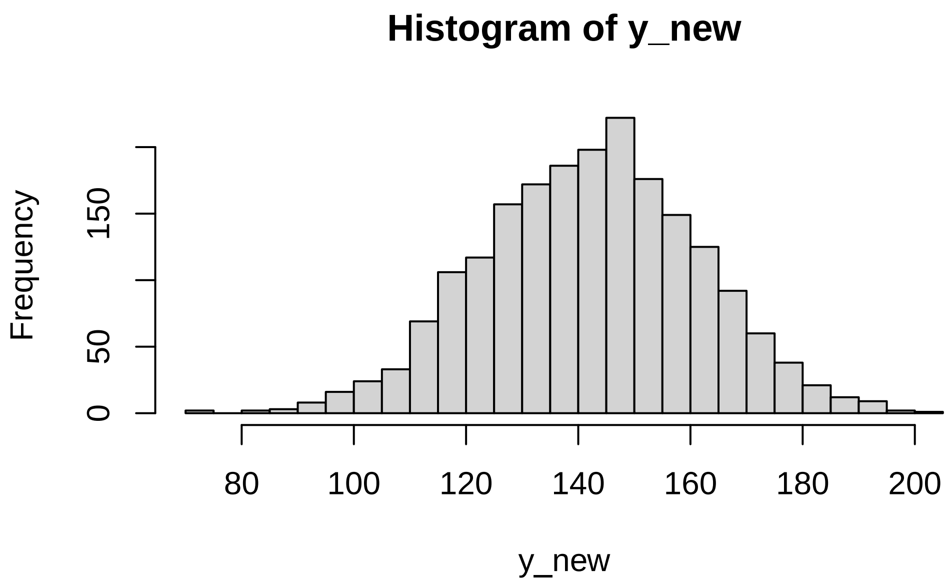
Question 7

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

```
# Extract parameters from the stanfit object
post_samples <- extract(fit3)
names(post_samples)
```

```
[1] "alpha" "beta"  "sigma" "lp__"
```

```
alpha <- post_samples[["alpha"]]
beta1 <- post_samples[["beta"]][,1]
beta2 <- post_samples[["beta"]][,2]
sigma <- post_samples[["sigma"]]
lin_pred <- alpha + beta1*1 + beta2*95
y_new <- rnorm(n = length(sigma), mean = lin_pred, sd = sigma)
hist(y_new, breaks=30)
```


Histogram of y_new
Frequency
0
50
150
80
100
120
140
160
180
200
y_new