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 1.3.2 --
## v ggplot2 3.4.0     v purrr   1.0.1
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.2.1     v stringr 1.5.0
## v readr   2.1.3     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(rstan)
```

```
## Loading required package: StanHeaders
##
## rstan version 2.26.15 (Stan version 2.26.1)
##
## 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)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
##
## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
##
## Attaching package: 'rstan'
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

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

```
## here() starts at C:/Users/nigel/OneDrive/School/First Year Masters/STA2201
```

The data look like this:

```
# kidiq <- read_rds(here("data","kidiq.RDS"))
kidiq <- readRDS("C:/Users/nigel/OneDrive/School/First Year Masters/STA2201/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

Estimating mean, no covariates

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

```
y <- kidiq$kid_score
mu0 <- 80
sigma0 <- 10

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

Now we can run the model:

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

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.23 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: 0.008 seconds (Warm-up)
## Chain 1:                0.005 seconds (Sampling)
## Chain 1:                0.013 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 6e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 2: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 2: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 2: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 2: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 2: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 2: Iteration: 251 / 500 [ 50%]  (Sampling)
## Chain 2: Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 2: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 2: Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 2: Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 2: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.013 seconds (Warm-up)
## Chain 2:                0.004 seconds (Sampling)
## Chain 2:                0.017 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: 0.012 seconds (Warm-up)
## Chain 3:                0.004 seconds (Sampling)
## Chain 3:                0.016 seconds (Total)
## Chain 3:
```

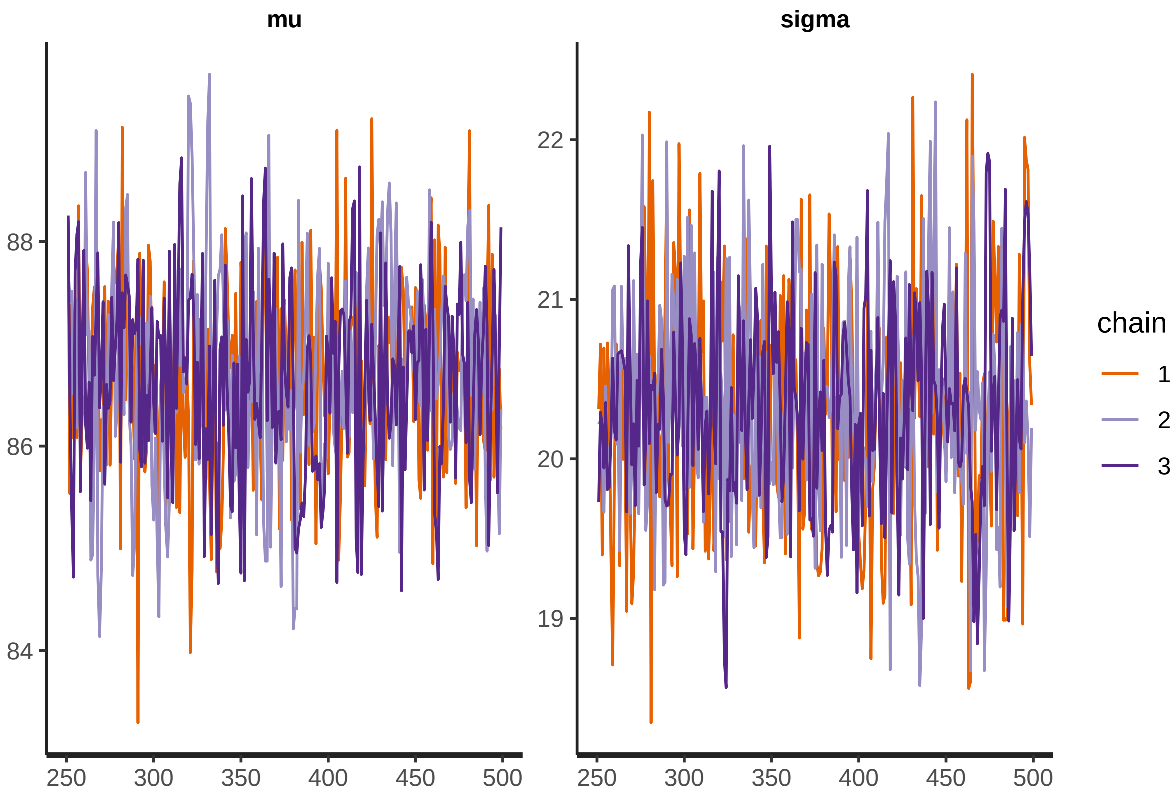
Look at the summary

```
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        86.68    0.04 0.97    84.77    86.04    86.69    87.34    88.52   561
## sigma     20.32    0.03 0.70    19.00    19.85    20.31    20.75    21.80   635
## lp__   -1525.76    0.05 1.01 -1528.58 -1526.15 -1525.46 -1525.04 -1524.78   376
##        Rhat
## mu        1
## sigma     1
## lp__      1
##
## Samples were drawn using NUTS(diag_e) at Mon Feb 13 22:32:02 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

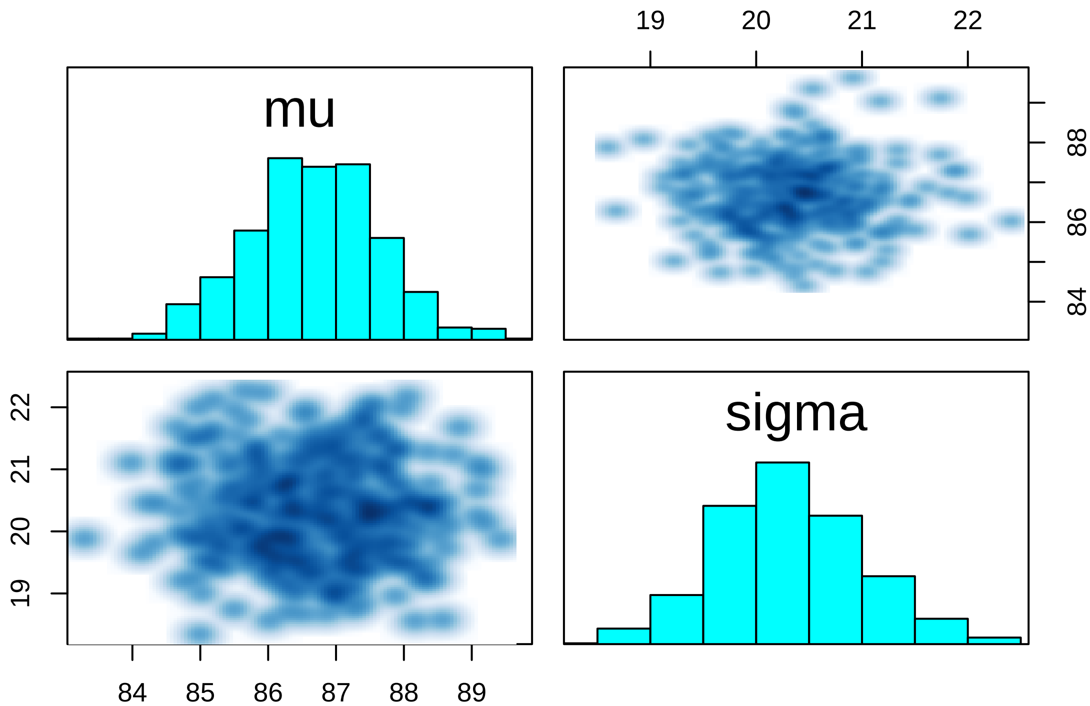
Traceplot

```
traceplot(fit)
```

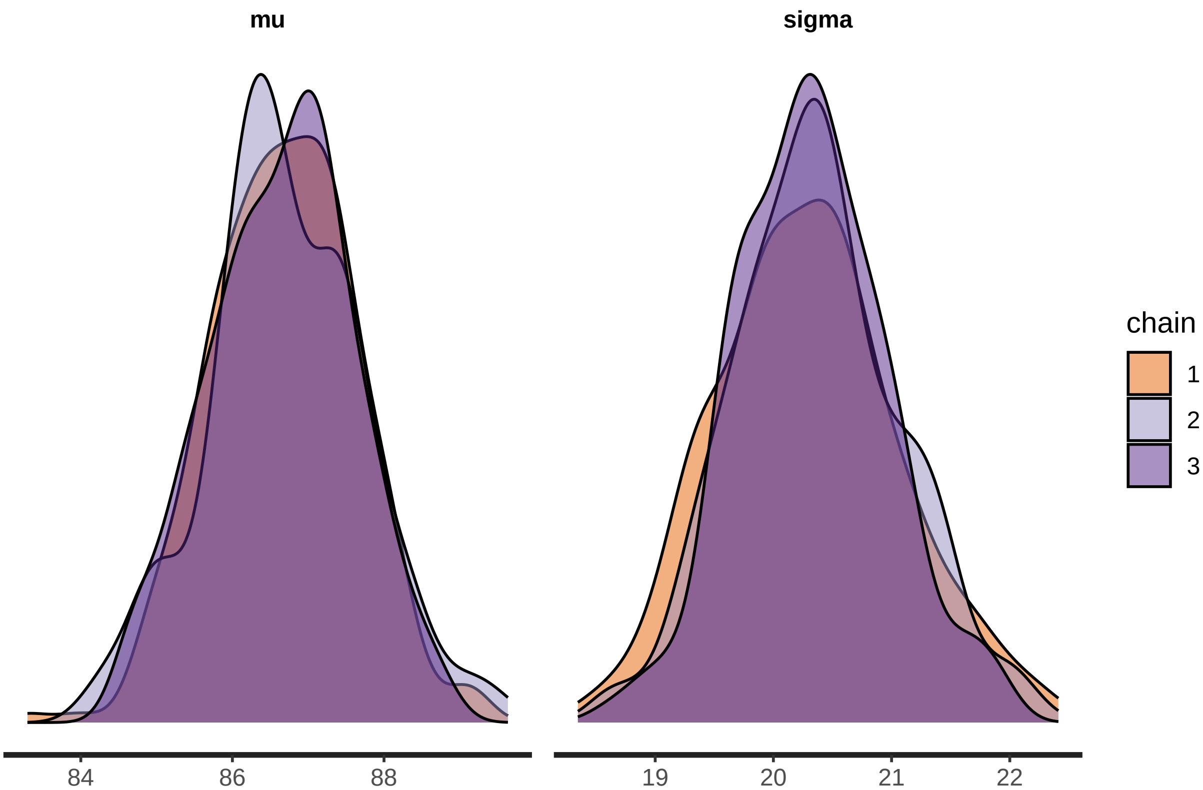


All looks fine.

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




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



Understanding output

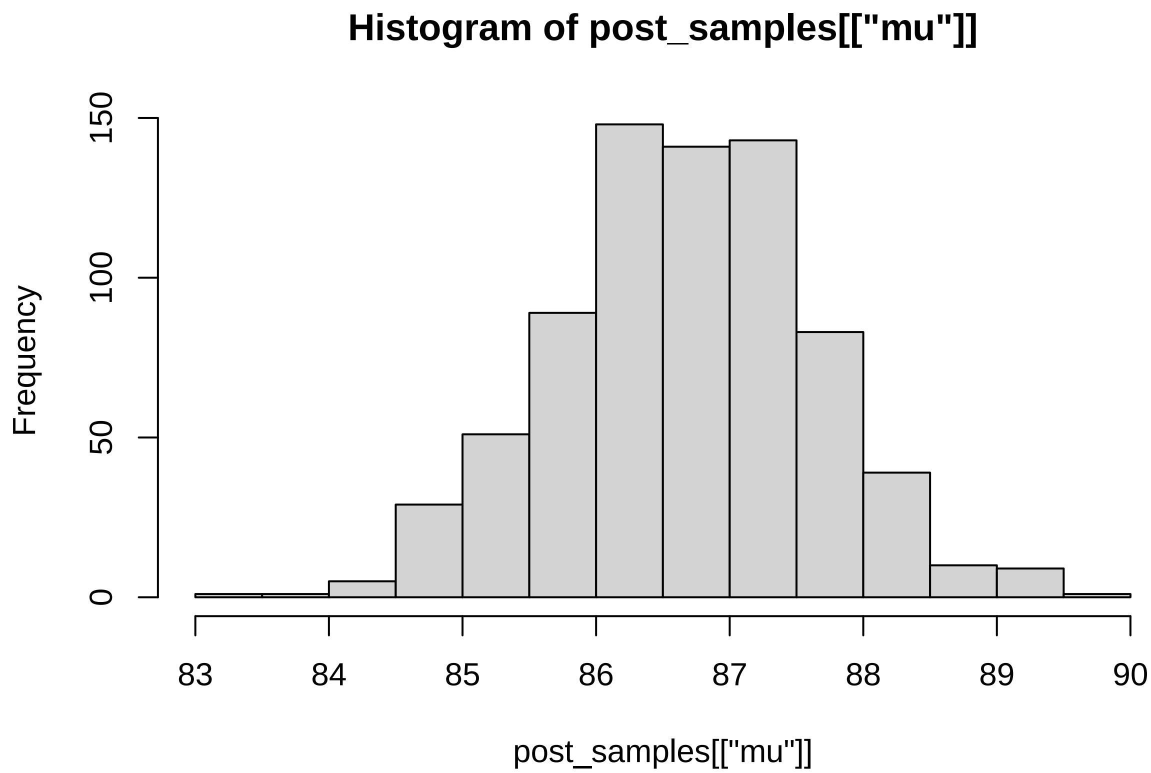
What does the model actually give us? A number of samples from the posteriors. To see this, we can use `extract` to get the samples.

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

```
## [1] 88.10911 86.07749 86.16805 86.28293 87.96373 87.12994
```

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] 86.68661
```

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

```
##     2.5%
## 84.76966
```

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

```
##    97.5%
## 88.52108
```

Plot estimates

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

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

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

```
## # A tibble: 1,500 x 5
## # Groups:   .variable [2]
##    .chain .iteration .draw .variable .value
##     <int>      <int> <int> <chr>      <dbl>
##  1      1          1     1 mu          88.2
##  2      1          2     2 mu          85.5
##  3      1          3     3 mu          87.4
##  4      1          4     4 mu          86.1
##  5      1          5     5 mu          87.2
##  6      1          6     6 mu          86.1
##  7      1          7     7 mu          88.3
##  8      1          8     8 mu          85.7
##  9      1          9     9 mu          86.3
## 10      1         10    10 mu          86.6
## # ... with 1,490 more rows
```

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

```
## # A tibble: 750 x 5
##    .chain .iteration .draw    mu sigma
##     <int>      <int> <int> <dbl> <dbl>
##  1      1          1     1  88.2  20.3
##  2      1          2     2  85.5  20.7
##  3      1          3     3  87.4  19.4
##  4      1          4     4  86.1  20.7
##  5      1          5     5  87.2  19.9
##  6      1          6     6  86.1  20.7
##  7      1          7     7  88.3  20.4
##  8      1          8     8  85.7  19.4
##  9      1          9     9  86.3  18.7
## 10      1         10    10  86.6  19.9
## # ... with 740 more rows
```

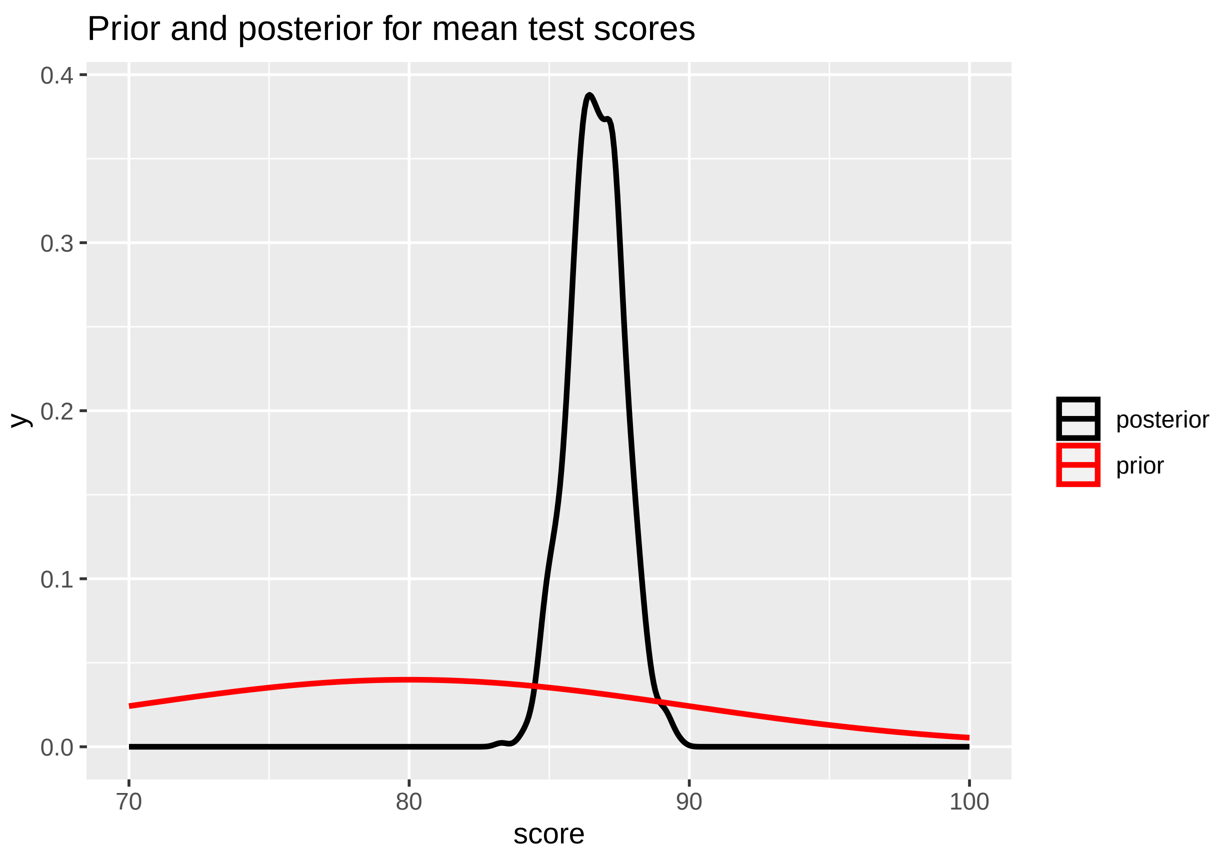
```
# quickly calculate the quantiles using

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   85.4   87.9    0.8 median qi
## 2 sigma       20.3   19.4   21.2    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.

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 = "kids3.stan",
            data = data,
            iter = 1000)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 9.1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.91 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: 0.097 seconds (Warm-up)
## Chain 1:                0.067 seconds (Sampling)
## Chain 1:                0.164 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.7e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.17 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: 0.153 seconds (Warm-up)
## Chain 2:                0.068 seconds (Sampling)
## Chain 2:                0.221 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: 0.135 seconds (Warm-up)
## Chain 3:                0.074 seconds (Sampling)
## Chain 3:                0.209 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.8e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.18 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: 0.165 seconds (Warm-up)
## Chain 4:                0.074 seconds (Sampling)
## Chain 4:                0.239 seconds (Total)
## Chain 4:
```

Question 3

a) Confirm that the estimates of the intercept and slope are comparable to results from `lm()`
b) Do a **pairs** plot to investigate the joint sample distributions of the slope and intercept. Comment briefly on what you see. Is this potentially a problem?

```
# part (a)

summary(fit2)$summary
```

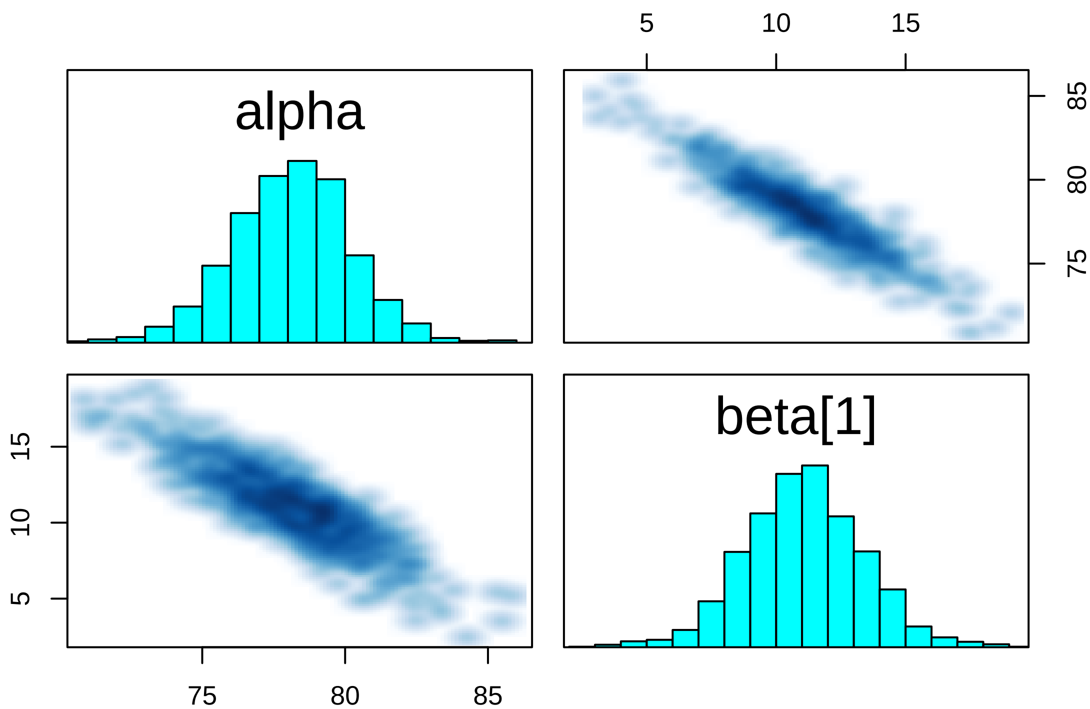
```
##                 mean    se_mean        sd         2.5%          25%          50%
## alpha       78.11739 0.08535450 2.0939807    73.918055    76.728153    78.19702
## beta[1]     11.07616 0.09245271 2.2950557     6.714268     9.574566    11.06342
## sigma       19.84022 0.02151720 0.6683784    18.561182    19.382264    19.84458
## lp__     -1514.40723 0.05521751 1.3135720 -1517.967746 -1514.919998 -1514.07105
##                  75%       97.5%    n_eff     Rhat
## alpha       79.46177    82.13901 601.8559 1.005429
## beta[1]     12.53630    15.68614 616.2361 1.005200
## sigma       20.27314    21.20757 964.8800 1.001035
## lp__     -1513.47475 -1512.98329 565.9188 1.010912
```

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

```
##
## 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 coefficients are similar from the two models.

```
#part (b)
pairs(fit2, pars = c("alpha", "beta"))
```



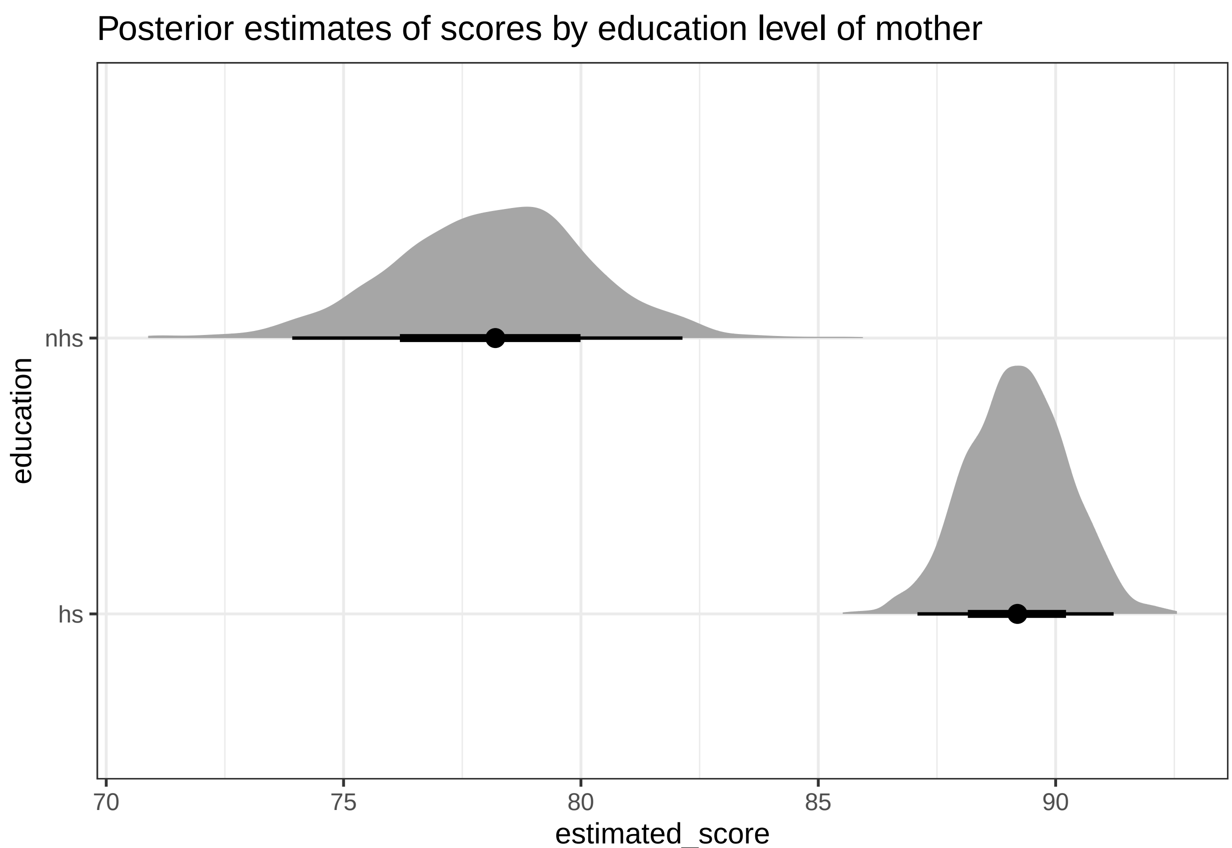
The scatter plots are fairly linear, suggesting a correlation between α and β.

Plotting results

It might be nice to plot the posterior samples of the estimates for the non-high-school and high-school mothered kids. Here's some code that does this: notice the `beta[condition]` syntax. Also notice I'm using `spread_draws`, because it's easier to calculate the estimated effects in wide format

```
fit2 |>
  spread_draws(alpha, beta[k], sigma) |>
     mutate(nhs = alpha, # no high school is just the intercept
          hs = alpha + beta) |>
  select(nhs, hs) |>
  pivot_longer(nhs:hs, names_to = "education", values_to = "estimated_score") |>
  ggplot(aes(y = education, x = estimated_score)) +
  stat_halfeye() +
  theme_bw() +
  ggtitle("Posterior estimates of scores by education level of mother")
```

```
## Adding missing grouping variables: ‘k‘
```



Question 4

Add in mother's IQ as a covariate and rerun the model. Please the mean center the covariate before putting it into the model. Interpret the coefficient on the (centered) mum's IQ.

```
# Test
X_new <- as.matrix(cbind(kidiq$mom_hs, kidiq$mom_iq - mean(kidiq$mom_iq)), ncol=2)
K <- 2

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

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.5e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.15 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: 0.121 seconds (Warm-up)
## Chain 1:                0.067 seconds (Sampling)
## Chain 1:                0.188 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.8e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.18 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: 0.119 seconds (Warm-up)
## Chain 2:                0.082 seconds (Sampling)
## Chain 2:                0.201 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.7e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.17 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: 0.119 seconds (Warm-up)
## Chain 3:                0.08 seconds (Sampling)
## Chain 3:                0.199 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.2 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: 0.114 seconds (Warm-up)
## Chain 4:                0.072 seconds (Sampling)
## Chain 4:                0.186 seconds (Total)
## Chain 4:
```

```
summary(fit3)$summary
```

```
##                  mean     se_mean         sd          2.5%           25%
## alpha      82.2300009 0.068147293 1.92515266    78.4021115    80.8904491
## beta[1]     5.7793482 0.078453575 2.21532614     1.4067507     4.2787925
## beta[2]     0.5641061 0.001773064 0.06241879     0.4379827     0.5220316
## sigma      18.1384444 0.016567076 0.64701846    16.8439523    17.7103161
## lp__    -1474.5590827 0.054689225 1.52882553 -1478.3258741 -1475.2960950
##                  50%           75%         97.5%     n_eff     Rhat
## alpha      82.257378    83.5104769    86.0814700  798.0556 1.000098
## beta[1]     5.760224     7.3421992    10.1836013  797.3516 1.000460
## beta[2]     0.565715     0.6054327     0.6847341 1239.3149 1.000181
## sigma      18.121676    18.5685280    19.4711319 1525.2521 1.000814
## lp__    -1474.188467 -1473.4427686 -1472.6566262  781.4700 1.004244
```

for every unit increase in the difference between the moms iq and the average iq, the childs test score will increase by 0.5667 points, with all other variables held fixed.

Question 5

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

```
model_2 = lm(kid_score~mom_hs+ I(mom_iq - mean(mom_iq)), data = kidiq)
summary(model_2)
```

```
##
## Call:
## lm(formula = kid_score ~ mom_hs + I(mom_iq - mean(mom_iq)), 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 **
## I(mom_iq - mean(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
```

The coefficients seem to be similar.

Question 6

Plot the posterior estimates of scores by education 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.