Lab exercise 4

Chenxi Liu 1010615050

2024-02-08

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

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(rstan)
```

```
## Loading required package: StanHeaders
##
## rstan version 2.32.5 (Stan version 2.32.2)
##
## 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)
##
##
## Attaching package: 'rstan'
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

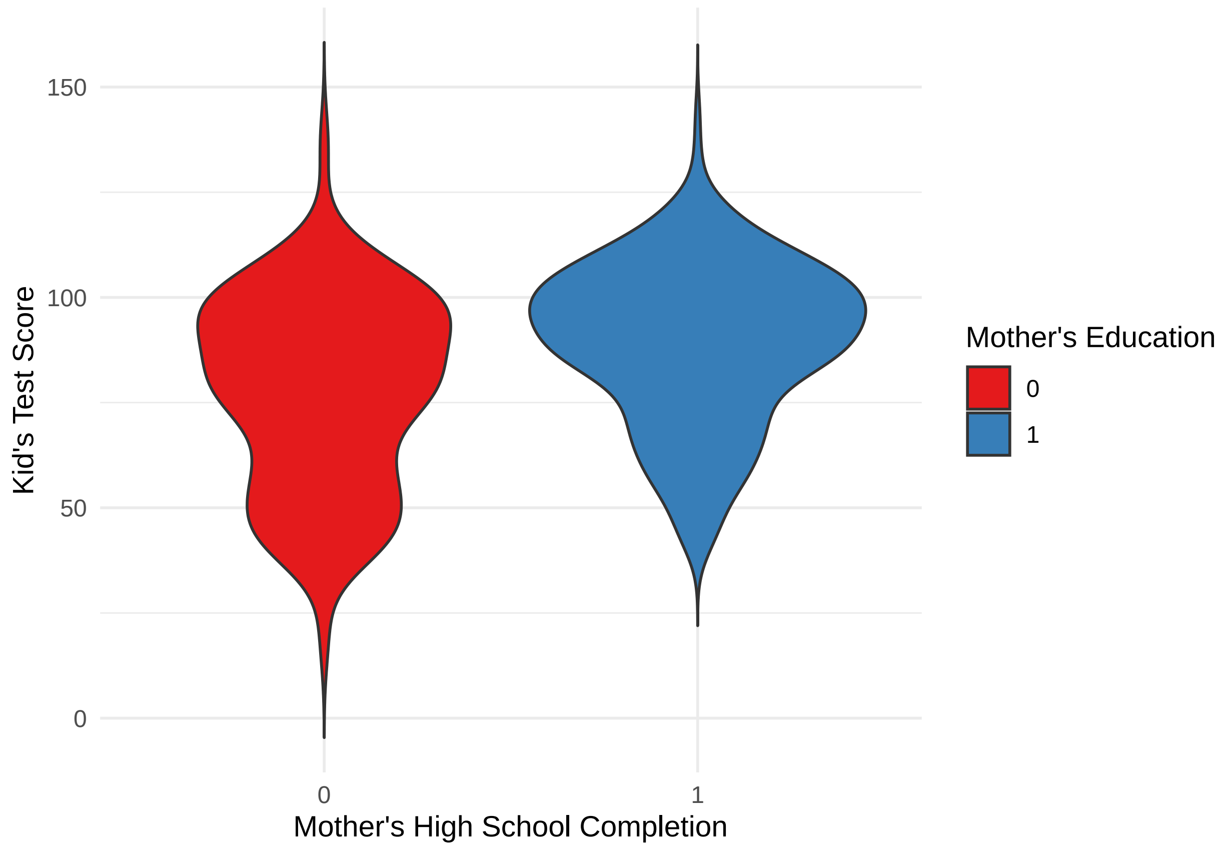
```r
library(tidybayes)
library(here)
```

```
## here() starts at /Users/dawn/Desktop/uoft/sta2201/HW
```

```r
kidiq <- readRDS("/Users/dawn/Desktop/uoft/sta2201/HW/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
## # i 424 more rows
```

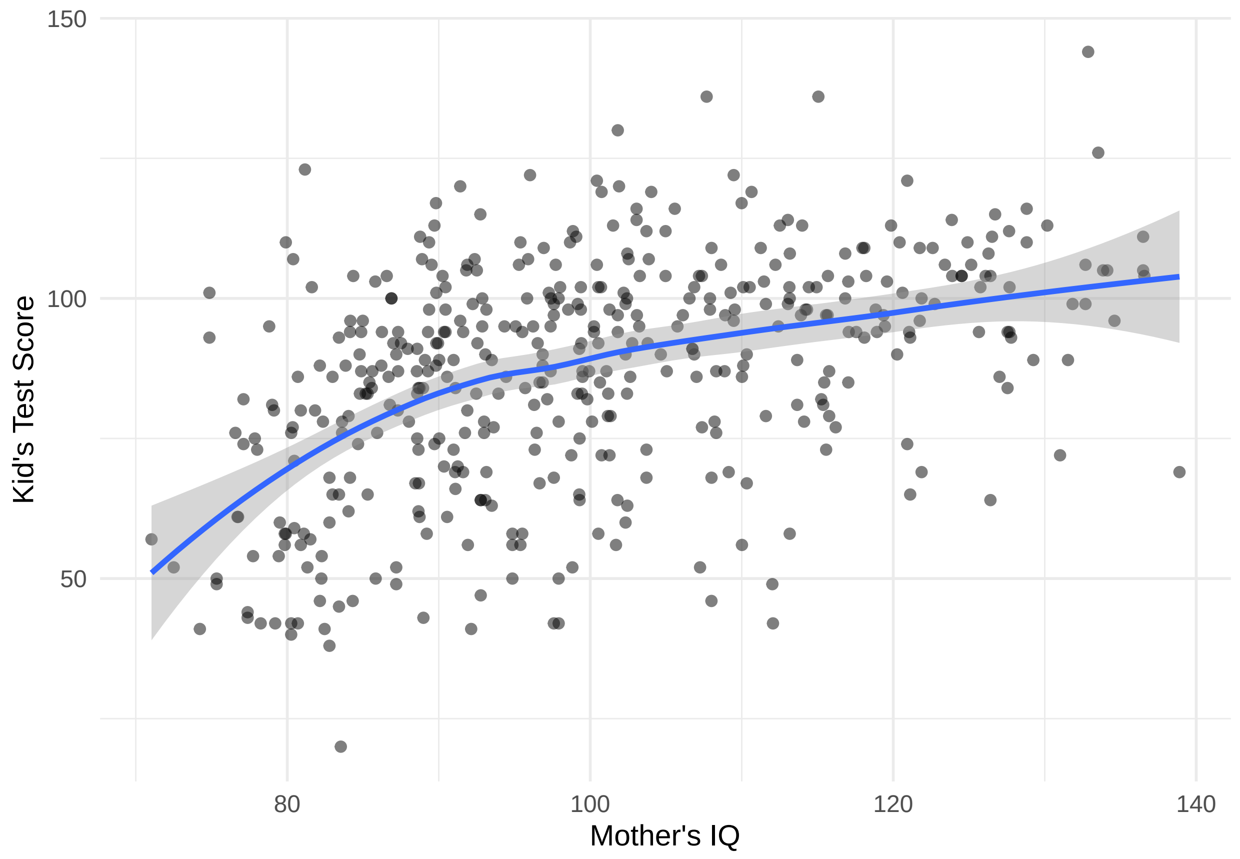
```r
library(ggplot2)
kidiq %>%
  ggplot(aes(x = factor(mom_hs), y = kid_score, fill = factor(mom_hs))) +
  geom_violin(trim = FALSE) +
  scale_fill_brewer(palette = "Set1", name = "Mother's Education") +
  labs(x = "Mother's High School Completion", y = "Kid's Test Score", fill = "Mother's Education") +
  theme_minimal()
```



Violin plots combine box plots with kernel density estimation, showing the distribution shape of test scores for each group. *The red violin (mother's education = 0) appears to be narrower and more pointed at both ends, which might suggest a slightly more uniform distribution with less variability than the blue violin (mother's education = 1).* The blue violin (mother's education = 1) seems to have a broader distribution, indicating that the kids' test scores vary more within this group. It also appears to be slightly shifted upwards, suggesting that children whose mothers completed high school tend to have higher test scores on average.

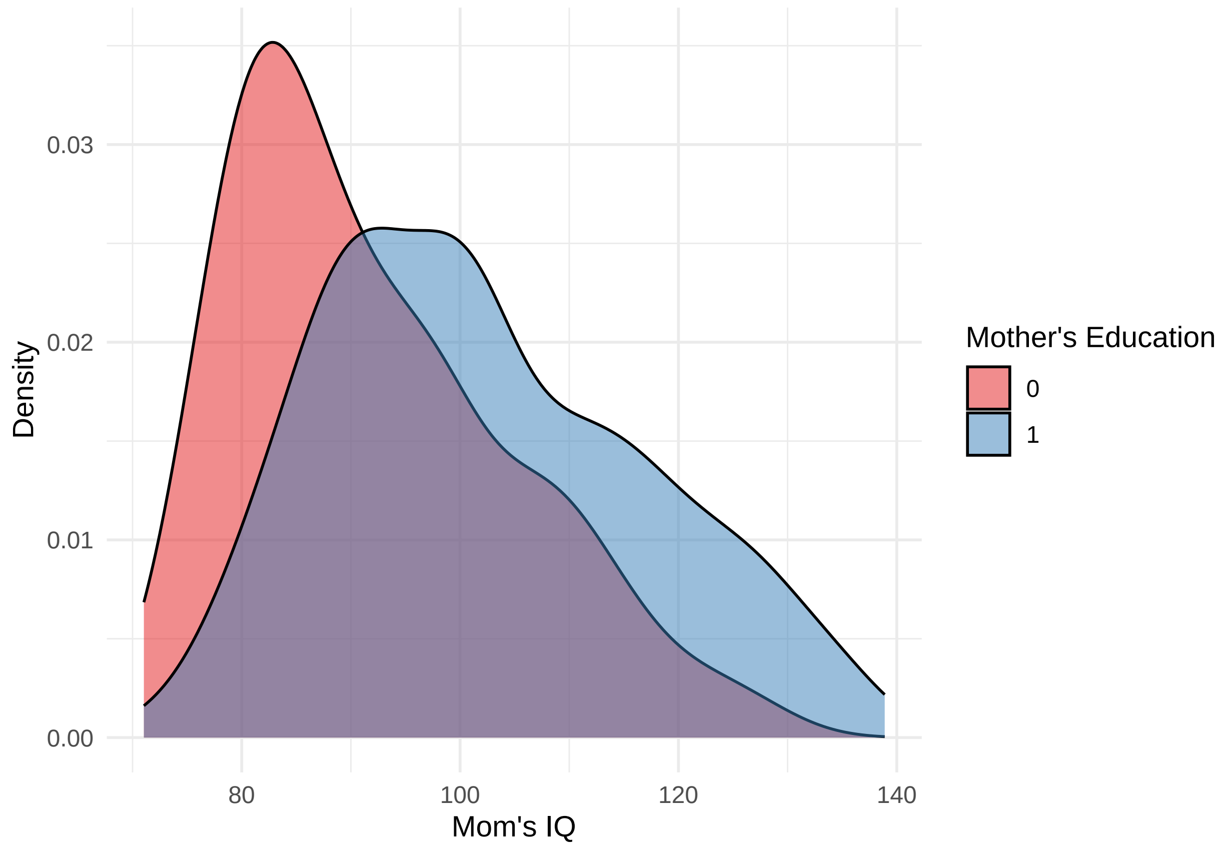
```r
kidiq %>%
  ggplot(aes(x = mom_iq, y = kid_score)) +
  geom_point(alpha = 0.5) +  # For scatter plot
  geom_smooth() +  # For density estimation
  labs(x = "Mother's IQ", y = "Kid's Test Score") +
  theme_minimal()
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



There appears to be a positive relationship between mother's iq and kid's score.There is considerable variability in the kid's test scores for any given level of mother's IQ. The shaded area around the fitted line represents a confidence interval for the mean response. The fact that this interval widens as mother's IQ increases suggests that the prediction of the kid's test score is less certain at higher levels of mother's IQ.

```r
kidiq %>%
  ggplot(aes(x = mom_iq, fill = as.factor(mom_hs))) +
  geom_density(alpha = 0.5) +
  scale_fill_brewer(palette = "Set1", name = "Mother's Education") +
  labs(x = "Mom's IQ", y = "Density") +
  theme_minimal()
```



It shows that in general, mom who have higher education level tend to show higher probability of high IQ, which is in line with the reality.

Estimating mean, no covariates

```
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)

fit <- stan(file = "/Users/dawn/Desktop/uoft/sta2201/HW/kids2.stan",
            data = data,
            chains = 3,
            iter = 500)
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line found on
## '/Users/dawn/Desktop/uoft/sta2201/HW/kids2.stan'

## Trying to compile a simple C file
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: ‘Apple clang version 14.0.3 (clang-1403.0.22.14.1)’
## using SDK: ‘MacOSX13.3.sdk’
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Framew
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/src/Core
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/src/Core
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/Core:96
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 6e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.06 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: 0.007 seconds (Warm-up)
## Chain 1:                0.002 seconds (Sampling)
## Chain 1:                0.009 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.01 seconds.
## Chain 2: Adjust your expectations accordingly!
```

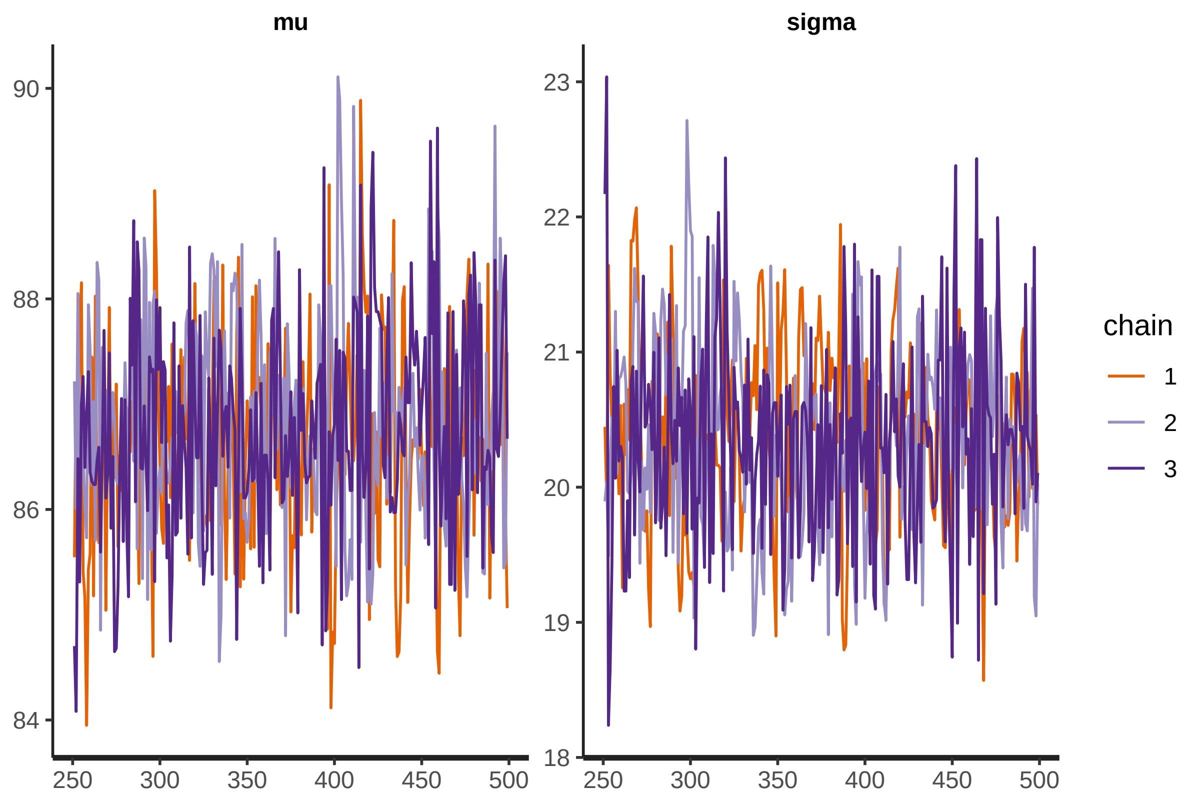
```
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 2: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 2: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 2: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 2: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 2: Iteration: 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: 0.004 seconds (Warm-up)
## Chain 2:                0.002 seconds (Sampling)
## Chain 2:                0.006 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.01 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: 0.004 seconds (Warm-up)
## Chain 3:                0.001 seconds (Sampling)
## Chain 3:                0.005 seconds (Total)
## Chain 3:
```

```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
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    0.04 0.99    84.83    86.08    86.68    87.38    88.59   541
## sigma    20.37    0.03 0.71    19.05    19.89    20.38    20.82    21.80   436
## lp__  -1525.80    0.06 1.07 -1528.66 -1526.27 -1525.49 -1525.03 -1524.78   293
##       Rhat
## mu    1.01
## sigma 1.00
## lp__  1.00
##
## Samples were drawn using NUTS(diag_e) at Fri Feb 16 13:03:12 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

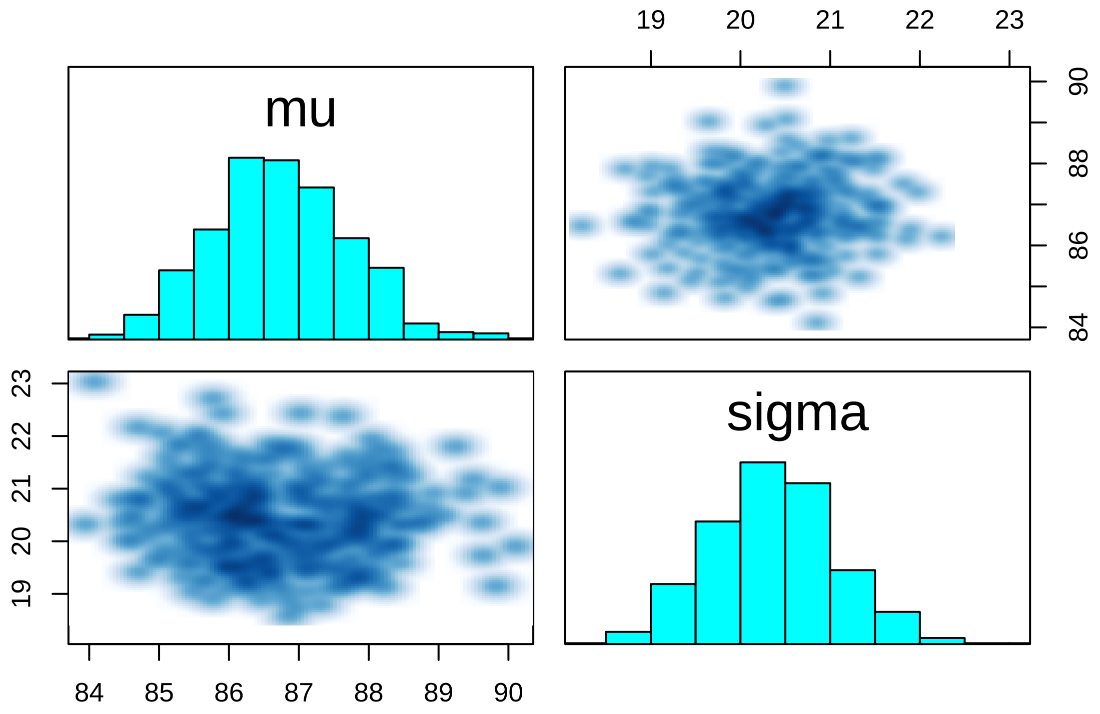
```r
traceplot(fit)
```



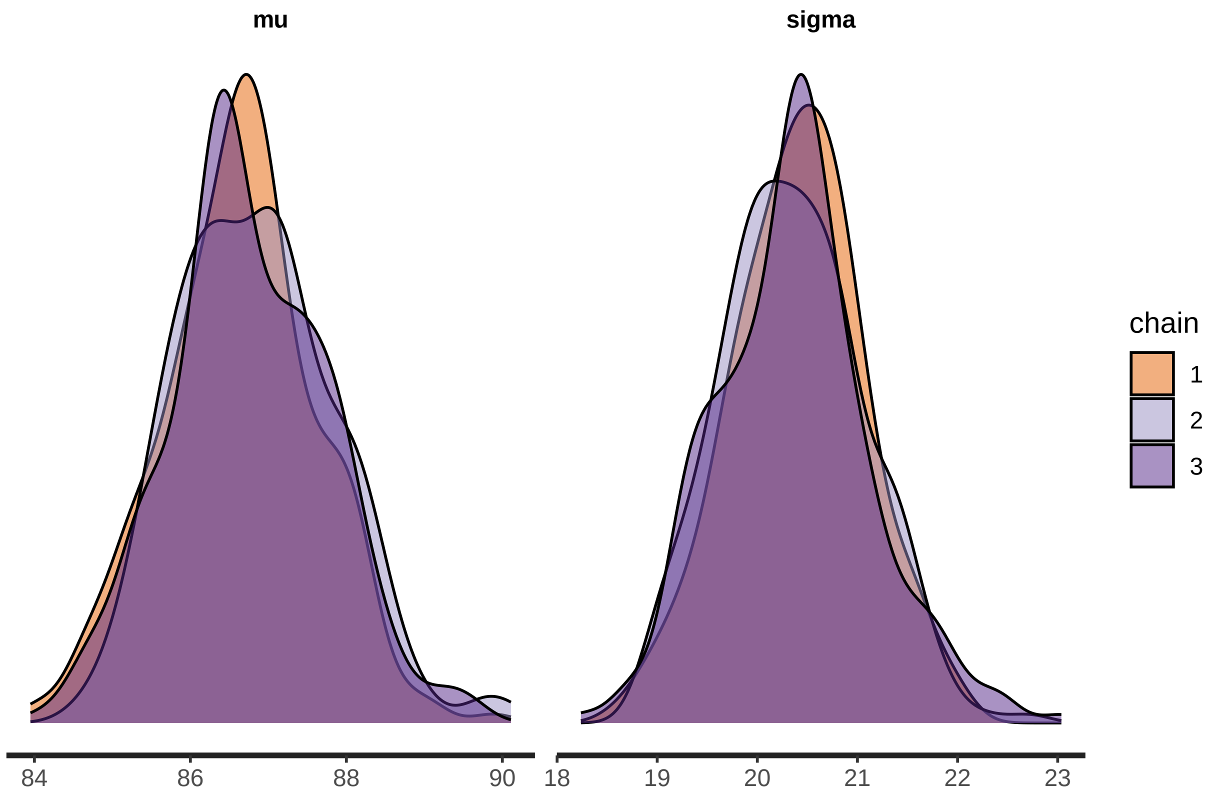
```r
pairs(fit, pars = c("mu", "sigma"))
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
```



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



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

```
## [1] 86.71211 86.11236 86.43910 88.12568 86.91194 84.44399
```

```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          85.5
##  2      1          2     2 mu          87.2
##  3      1          3     3 mu          86.8
##  4      1          4     4 mu          87.6
##  5      1          5     5 mu          88.2
##  6      1          6     6 mu          85.4
##  7      1          7     7 mu          85.2
##  8      1          8     8 mu          83.9
##  9      1          9     9 mu          85.4
## 10      1         10    10 mu          85.6
## # i 1,490 more rows
```

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

```
## # A tibble: 750 x 5
##    .chain .iteration .draw    mu sigma
##     <int>      <int> <int> <dbl> <dbl>
##  1      1          1     1  85.5  20.4
##  2      1          2     2  87.2  19.9
##  3      1          3     3  86.8  21.6
##  4      1          4     4  87.6  20.8
##  5      1          5     5  88.2  20.5
##  6      1          6     6  85.4  20.4
##  7      1          7     7  85.2  20.3
##  8      1          8     8  83.9  20.3
##  9      1          9     9  85.4  20.0
## 10      1         10    10  85.6  20.6
## # i 740 more rows
```

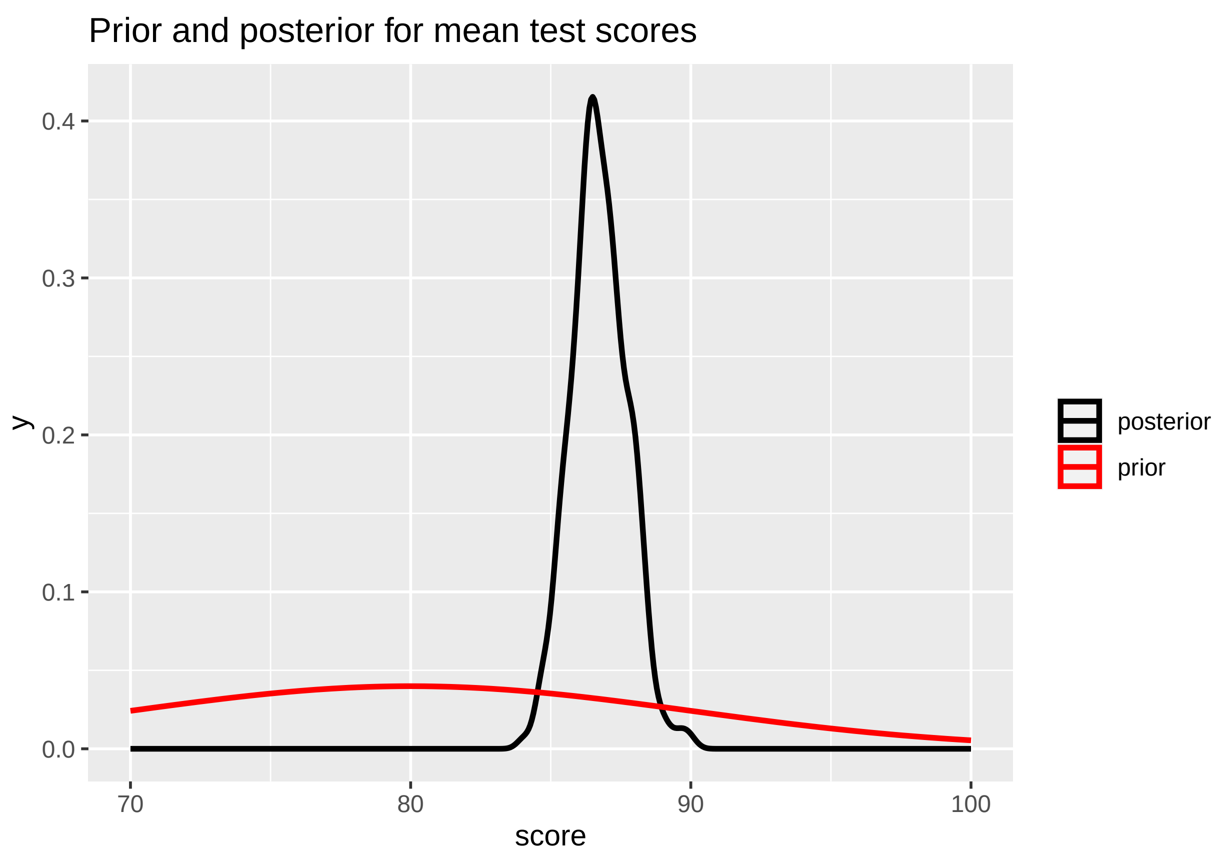
```r
# quickly calculate the quantiles using

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

```
## # A tibble: 2 x 7
##   .variable .value .lower .upper .width .point .interval
##   <chr>      <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
## 1 mu          86.7   85.5   88.0    0.8 median qi
## 2 sigma       20.4   19.4   21.3    0.8 median qi
```

```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")
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



## 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.

```
sigma0 <- 0.1

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

fit <- stan(file ="/Users/dawn/Desktop/uoft/sta2201/HW/kids2.stan",
            data = data)
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line found on
## '/Users/dawn/Desktop/uoft/sta2201/HW/kids2.stan'
```

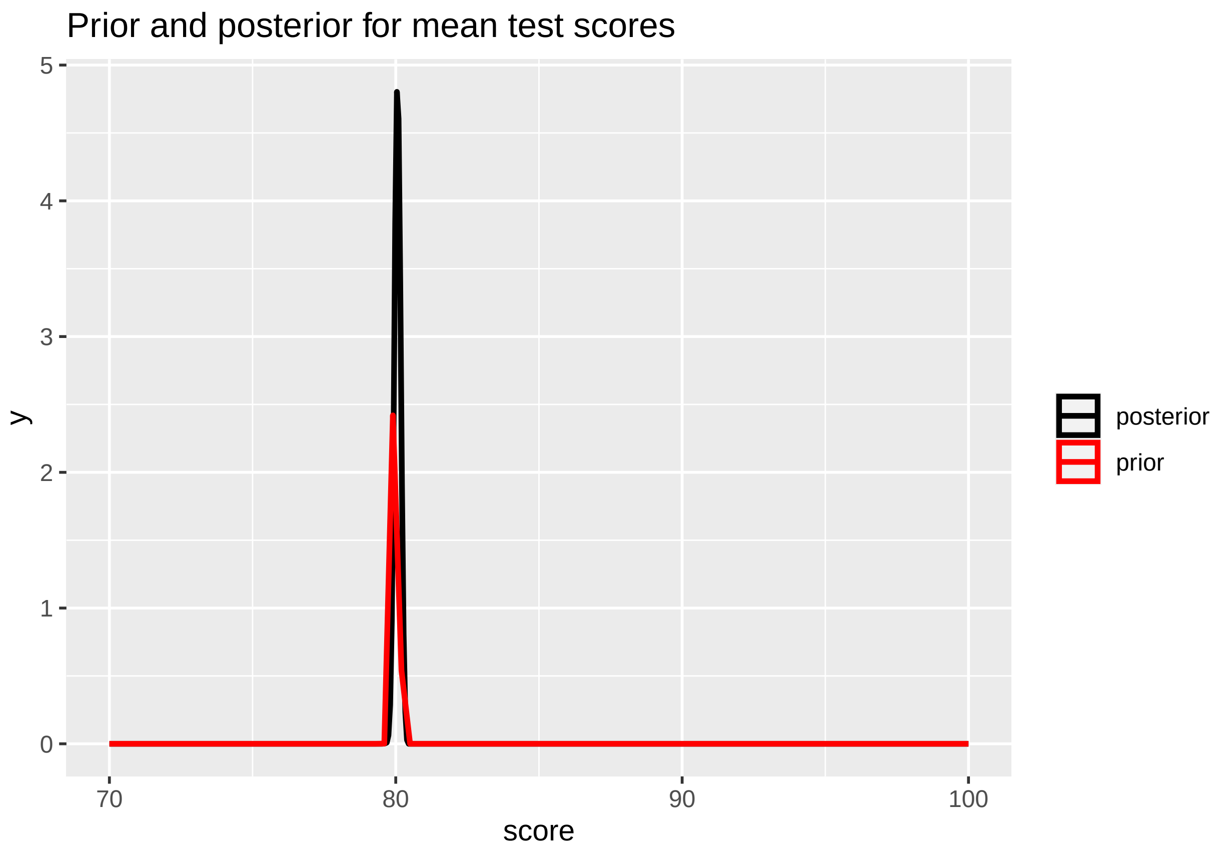
```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    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: 0.008 seconds (Warm-up)
## Chain 1:                0.007 seconds (Sampling)
## Chain 1:                0.015 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.02 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: 0.008 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 1e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.01 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: 0.007 seconds (Warm-up)
## Chain 3:                0.008 seconds (Sampling)
## Chain 3:                0.015 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.01 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: 0.008 seconds (Warm-up)
## Chain 4:                0.007 seconds (Sampling)
## Chain 4:                0.015 seconds (Total)
## Chain 4:
```

```
summary(fit)[["summary"]]
```

```
##               mean     se_mean        sd        2.5%         25%         50%
## mu        80.06265 0.001695312 0.1018951    79.87010    79.99226    80.06193
## sigma     21.41739 0.012345132 0.7219398    20.06922    20.91653    21.39858
## lp__   -1548.39289 0.021929198 0.9838621 -1550.97249 -1548.78202 -1548.10017
##                75%       97.5%    n_eff     Rhat
## mu        80.13283    80.26338 3612.496 1.000126
## sigma     21.89889    22.88494 3419.878 1.000908
## lp__   -1547.67658 -1547.39762 2012.904 1.001642
```

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

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")
```



```
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 = "/Users/dawn/Desktop/uoft/sta2201/HW/kids3.stan",
            data = data,
            iter = 1000)
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line found on
## '/Users/dawn/Desktop/uoft/sta2201/HW/kids3.stan'
```

```
## Trying to compile a simple C file
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: ‘Apple clang version 14.0.3 (clang-1403.0.22.14.1)’
## using SDK: ‘MacOSX13.3.sdk’
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Frame
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/src/Cor
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/src/Cor
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/Core:96
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.42 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: 0.069 seconds (Warm-up)
## Chain 1:                0.041 seconds (Sampling)
## Chain 1:                0.11 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 9e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Chain 2: Adjust your expectations accordingly!
```

```
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 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: 0.052 seconds (Warm-up)
## Chain 2:                0.033 seconds (Sampling)
## Chain 2:                0.085 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 8e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:   1 / 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: 0.072 seconds (Warm-up)
## Chain 3:                0.041 seconds (Sampling)
## Chain 3:                0.113 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 8e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.08 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: 0.053 seconds (Warm-up)
## Chain 4:                0.037 seconds (Sampling)
## Chain 4:                0.09 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?

```
summary(fit2)$summary[1:2,]
```

```
##             mean    se_mean       sd      2.5%       25%      50%      75%
## alpha   78.09912 0.07008176 1.963977 74.318885 76.766608 78.04535 79.41170
## beta[1] 11.05233 0.08570413 2.196841  6.629384  9.630046 11.05788 12.52806
##            97.5%    n_eff     Rhat
## alpha   82.10993 785.3493 1.002697
## beta[1] 15.45465 657.0431 1.001776
```

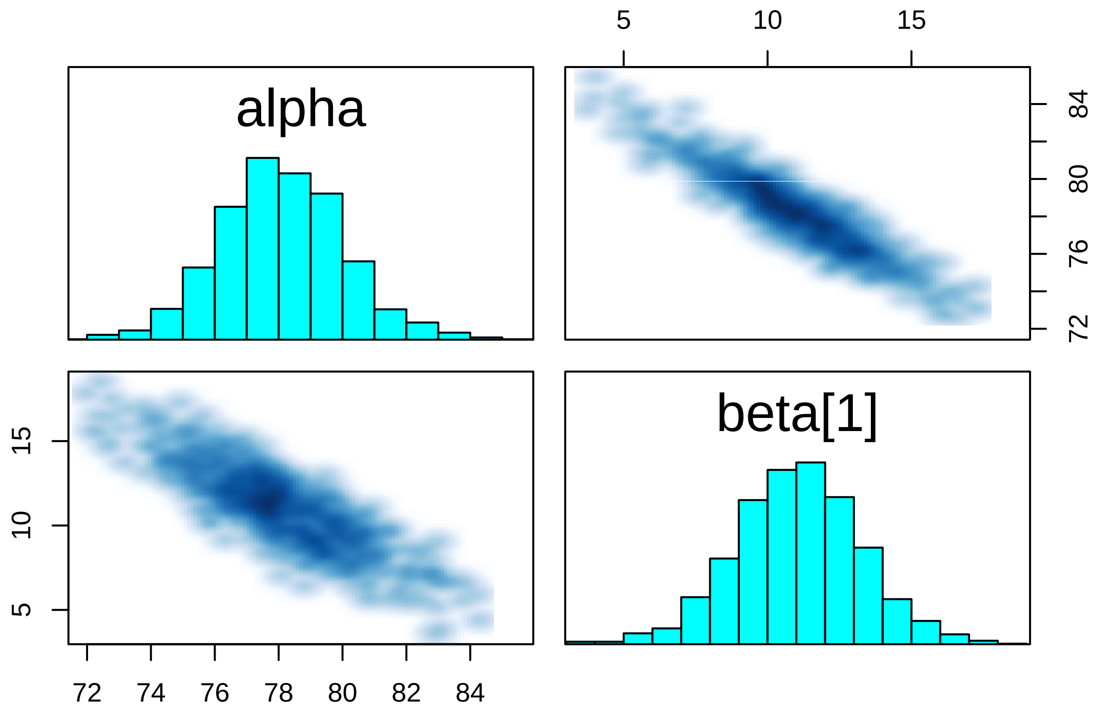
```
summary(lm(y~kidiq$mom_hs))
```

```
##
## Call:
## lm(formula = y ~ kidiq$mom_hs)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -57.55 -13.32   2.68  14.68  58.45
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    77.548      2.059  37.670  < 2e-16 ***
## kidiq$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
```

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

```
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
```



There is strong linear relationship, which indicates a high correlation between parameters.If the slope and intercept are highly correlated, it can make the model sensitive to the scale of the predictor variable.

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.

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

X <- cbind(kidiq$mom_hs, kidiq$mom_iq - mean(kidiq$mom_iq))
K <- 2

data <- list(y = y, N = length(y),
             X =X, K = K)
fit <- stan(file = "/Users/dawn/Desktop/uoft/sta2201/HW/kids3.stan",
            data = data,
            iter = 1000)
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line found on
## '/Users/dawn/Desktop/uoft/sta2201/HW/kids3.stan'

##
## 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: 0.067 seconds (Warm-up)
## Chain 1:                0.049 seconds (Sampling)
## Chain 1:                0.116 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.1 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: 0.081 seconds (Warm-up)
## Chain 2:                0.048 seconds (Sampling)
## Chain 2:                0.129 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 8e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:   1 / 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: 0.098 seconds (Warm-up)
## Chain 3:                0.044 seconds (Sampling)
## Chain 3:                0.142 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 8e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.08 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: 0.07 seconds (Warm-up)
## Chain 4:                0.042 seconds (Sampling)
## Chain 4:                0.112 seconds (Total)
## Chain 4:
```

```r
fit
```

```
## 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       82.30    0.06 1.87    78.68    81.05    82.31    83.57    85.91
## beta[1]      5.74    0.06 2.10     1.67     4.29     5.79     7.19     9.85
## beta[2]      0.56    0.00 0.06     0.45     0.52     0.57     0.60     0.68
## sigma       18.12    0.02 0.61    16.99    17.70    18.10    18.52    19.44
## lp__     -1474.39    0.05 1.36 -1477.74 -1475.10 -1474.11 -1473.40 -1472.67
##          n_eff Rhat
## alpha     1089    1
## beta[1]   1155    1
## beta[2]   1360    1
## sigma     1337    1
## lp__       737    1
##
## Samples were drawn using NUTS(diag_e) at Fri Feb 16 13:03:32 2024.
## 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).
```

beta[2]: The coefficient for the centered mom_iq variable. The mean value is 0.57, with a standard error of 0.00, and a standard deviation of 0.06. The 95% credible interval for the coefficient is between 0.45 and 0.69. The mean of coefficient of mom's iq is positive which shows a positive association with kids iq, this means that for a one-unit increase in the mum's IQ from its mean value, the kid's score is expected to increase by 0.57 units, holding all other variables constant.

Question 5

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

```r
lm_model <- lm(y ~ X[,1] + X[,2])
summary(lm_model)
```

```
##
## Call:
## lm(formula = y ~ X[, 1] + X[, 2])
##
## 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 ***
## X[, 1]       5.95012    2.21181   2.690  0.00742 **
## X[, 2]       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 obtained from the Bayesian model using Stan are indeed consistent with the coefficients obtained from the frequentist linear regression model using lm().

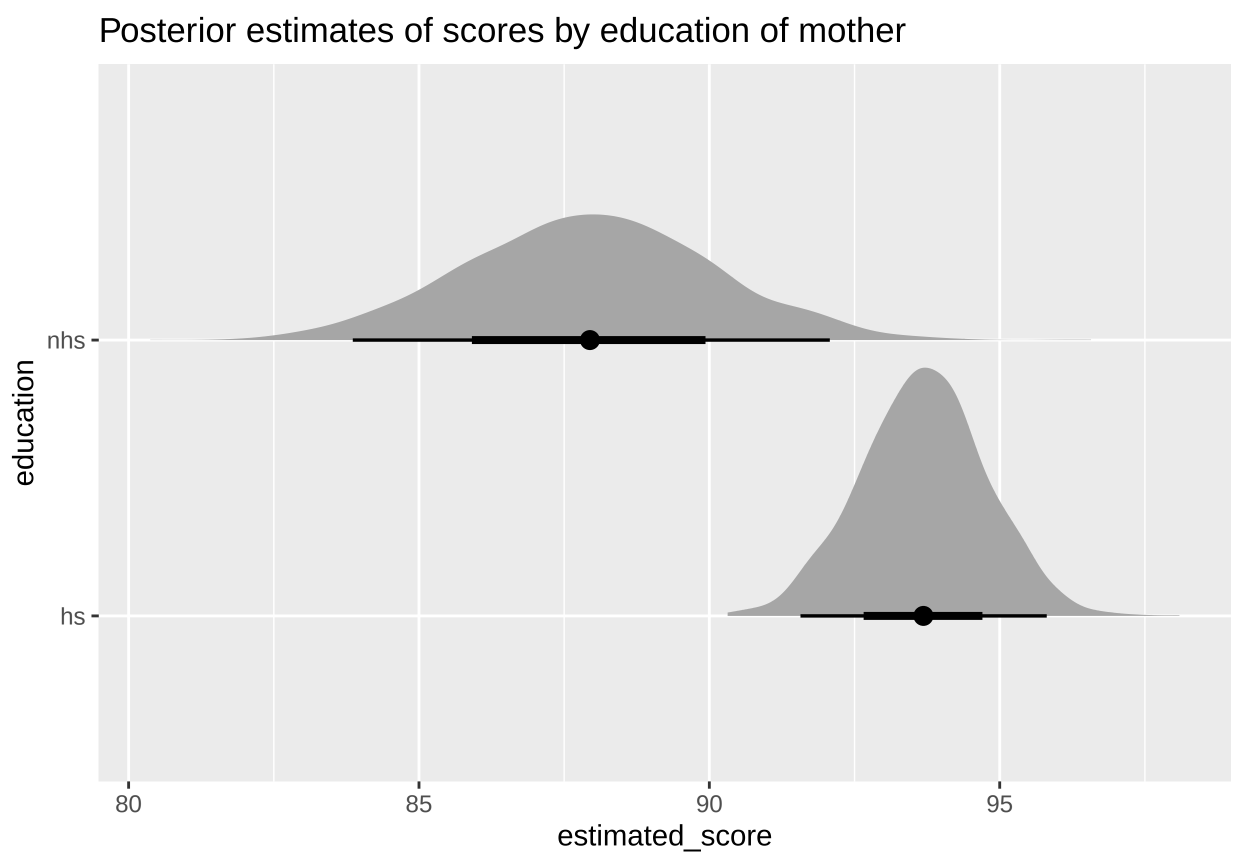
Question 6

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

```r
center_IQ <- 110-mean(kidiq$mom_iq) #which is 10

fit %>%
  gather_draws(alpha, beta[condition]) %>%
  group_by(.draw) %>%
  mutate(.value = ifelse(!is.na(condition)&condition==2, .value*center_IQ,  .value)) %>%
  summarise(nhs = sum(.value[is.na(condition)|condition==2]),
            hs = sum(.value)) %>%
  pivot_longer(nhs:hs, names_to = "education", values_to = "estimated_score") %>%
  ggplot(aes(y = education, x = estimated_score)) +
  stat_halfeyeh() +
  ggtitle("Posterior estimates of scores by education of mother")
```

```
## Warning: 'stat_halfeyeh' is deprecated.
## Use 'stat_halfeye' instead.
## See help("Deprecated") and help("tidybayes-deprecated").
```



Question 7

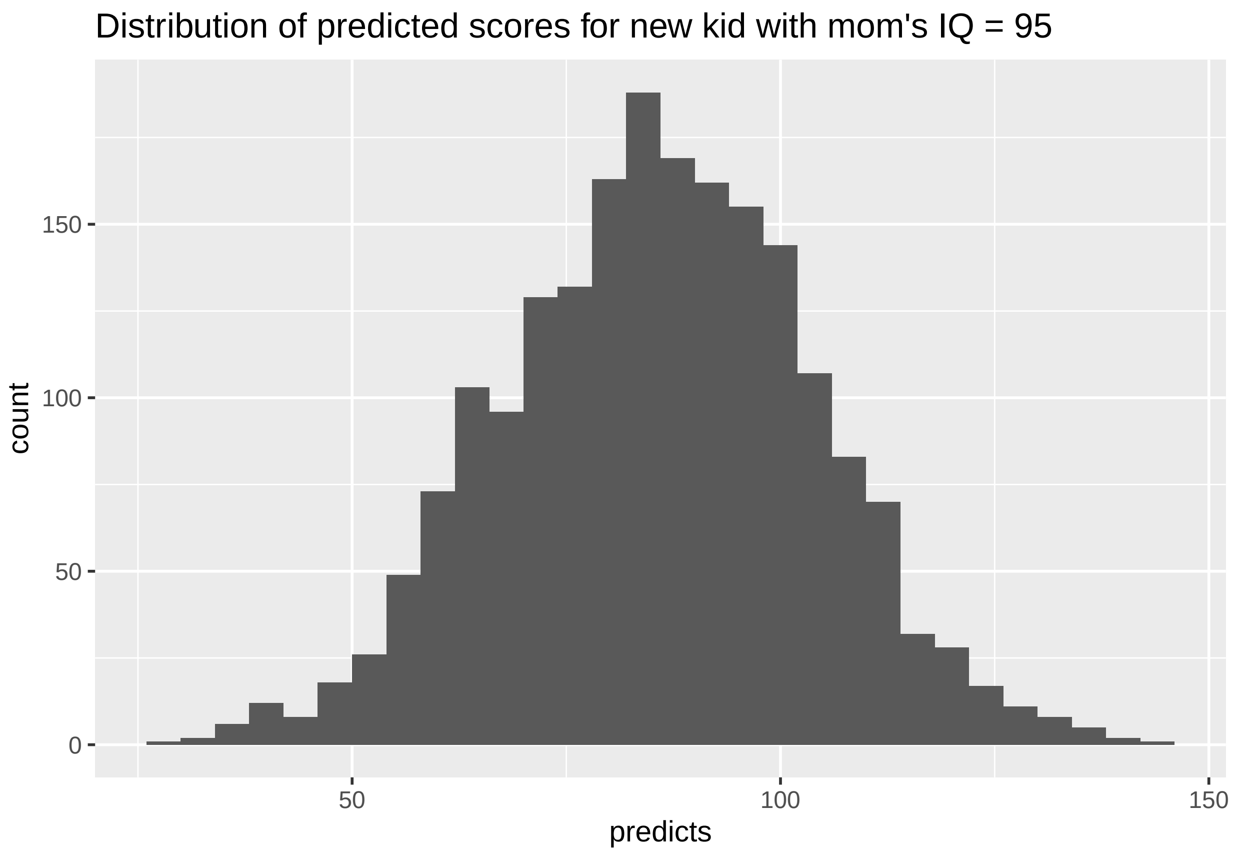
Generate and plot (as a histogram) samples from the posterior predictive distribution for a new kid with a mother who graduated high school and has an IQ of 95.

```
center_IQ <- 95-mean(kidiq$mom_iq)

samples <- rstan::extract(fit)
mu <- samples[["alpha"]] + samples[["beta"]][,1] + samples[["beta"]][,2]*center_IQ
sigmas <- samples[["sigma"]]

predicts <- tibble(predicts = rnorm(length(sigmas), mean = mu, sd = sigmas))
ggplot(predicts,aes(predicts)) + geom_histogram() + ggtitle("Distribution of predicted scores for new ki
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



Distribution of predicted scores for new kid with mom's IQ = 95
count
0
50
100
150
predicts
50
100
150