# Lab 6

Roxanne Li

2024-02-19

```
library(tidyverse)
library(here)
# for bayes stuff
library(rstan)
library(bayesplot)
library(loo)
library(tidybayes)
library(ggplot2)
```

Loading the data:

```
ds <- read_rds("data/births_2017_sample.RDS")
head(ds)
```

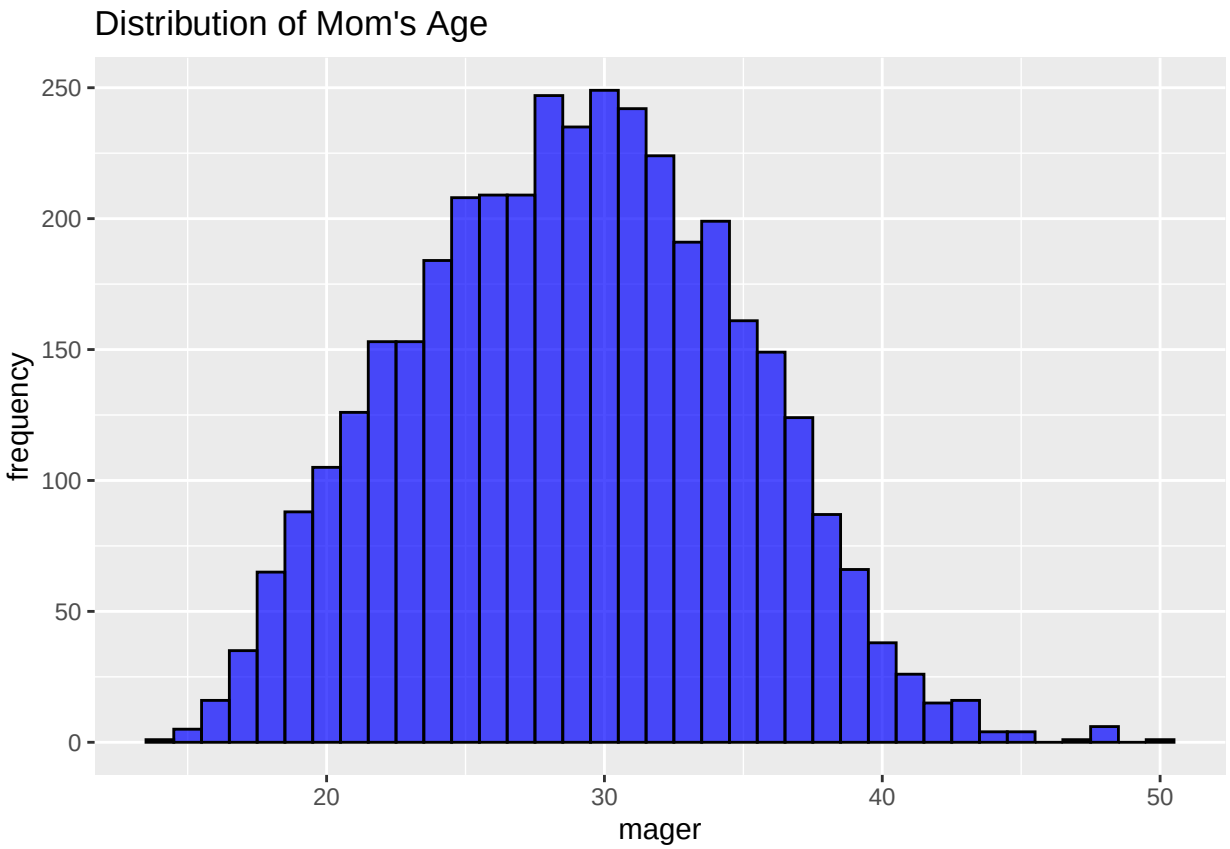
```
## # A tibble: 6 x 8
##   mager mracehisp meduc   bmi sex   combgest  dbwt ilive
##   <dbl>     <dbl> <dbl> <dbl> <chr>    <dbl> <dbl> <chr>
## 1    16         2     2  23   M           39  3.18 Y
## 2    25         7     2  43.6 M           40  4.14 Y
## 3    27         2     3  19.5 F           41  3.18 Y
## 4    26         1     3  21.5 F           36  3.40 Y
## 5    28         7     2  40.6 F           34  2.71 Y
## 6    31         7     3  29.3 M           35  3.52 Y
```

```
ds <- ds %>%
  rename(birthweight = dbwt, gest = combgest) %>%
  mutate(preterm = ifelse(gest<32, "Y", "N")) %>%
  filter(ilive=="Y",gest< 99, birthweight<9.999)
```

## Q1

Plot the distribution of mom's age:

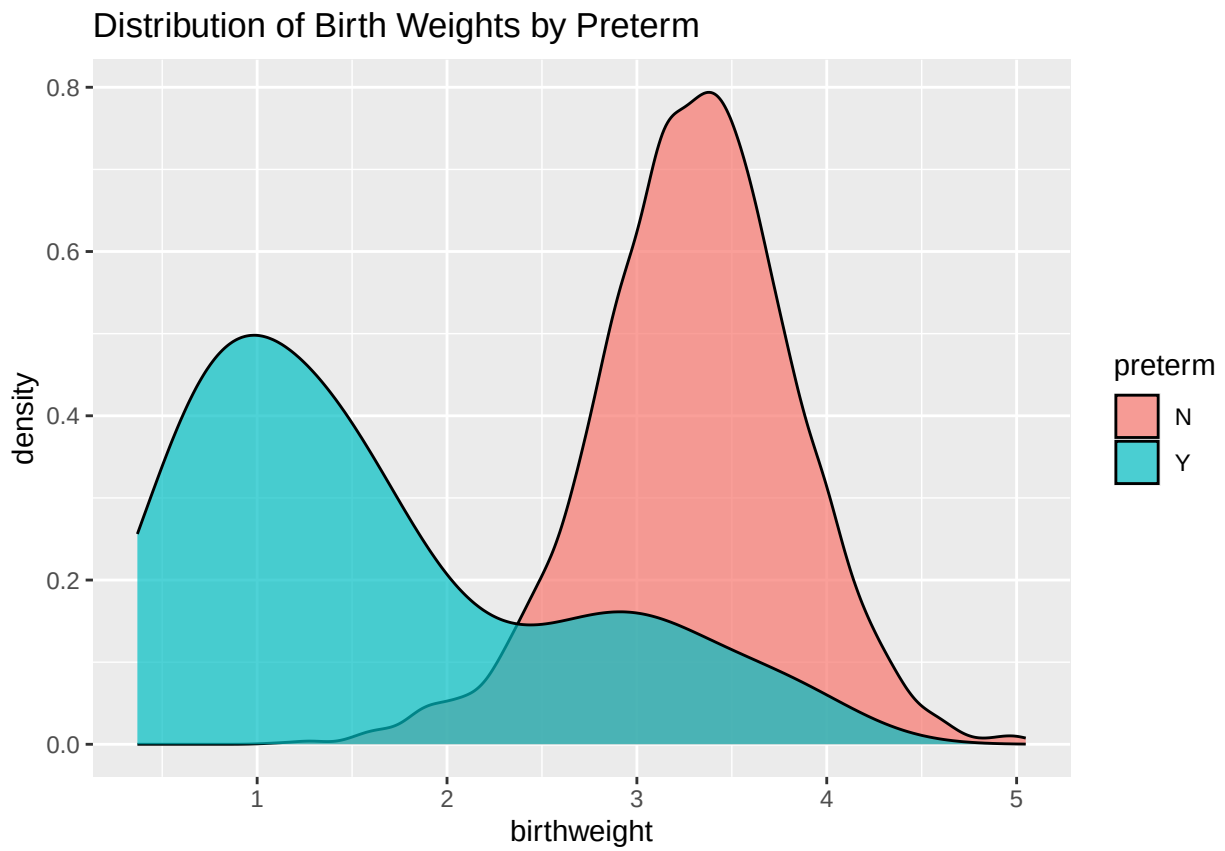
```
ggplot(ds, aes(x = mager)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Distribution of Mom's Age",
       x = "mager",
       y = "frequency")
```




The histogram of mom's age shows that the ages of moms are centered around 30, ranging from 14 to 50.

Plot the distributions of birth weights for preterm and non-preterm births:

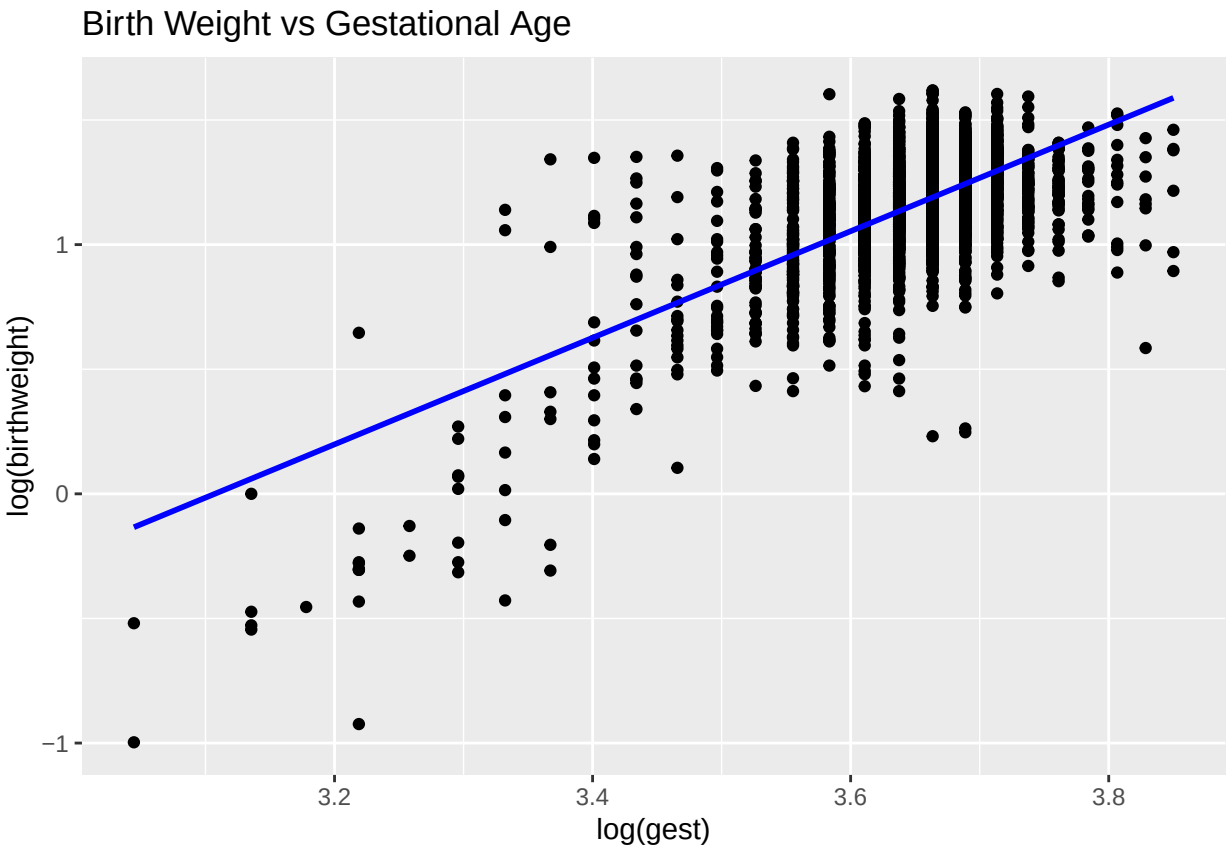
```
ggplot(ds, aes(x = birthweight, fill = preterm)) +
  geom_density(alpha = 0.7) +
  labs(title = "Distribution of Birth Weights by Preterm",
       x = "birthweight",
       y = "density",
       fill = "preterm")
```




The density plot shows that preterm births tend to have lower birth weights whereas non-preterm births tend to have higher birth weights.

Plot log of birth weight against log of gestational age, and add a linear fit line on the plot:

```
ggplot(ds, aes(x = log(gest), y = log(birthweight))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(title = "Birth Weight vs Gestational Age",
       x = "log(gest)",
       y = "log(birthweight)")
```




The scatter plot and the linear fit line shows that there is a positive linear relationship between the log of birth weight and the log of gestational age.

## Q2

Centering and standardizing gestational age:

```
ds$log_gest_c <- (log(ds$gest) - mean(log(ds$gest)))/sd(log(ds$gest))
```

Specifying priors:

```
set.seed(123)
N <- 1000

beta1 <- rnorm(N, mean = 0, sd = 1)
beta2 <- rnorm(N, mean = 0, sd = 1)
sigma <- abs(rnorm(N, mean = 0, sd = 1))
```

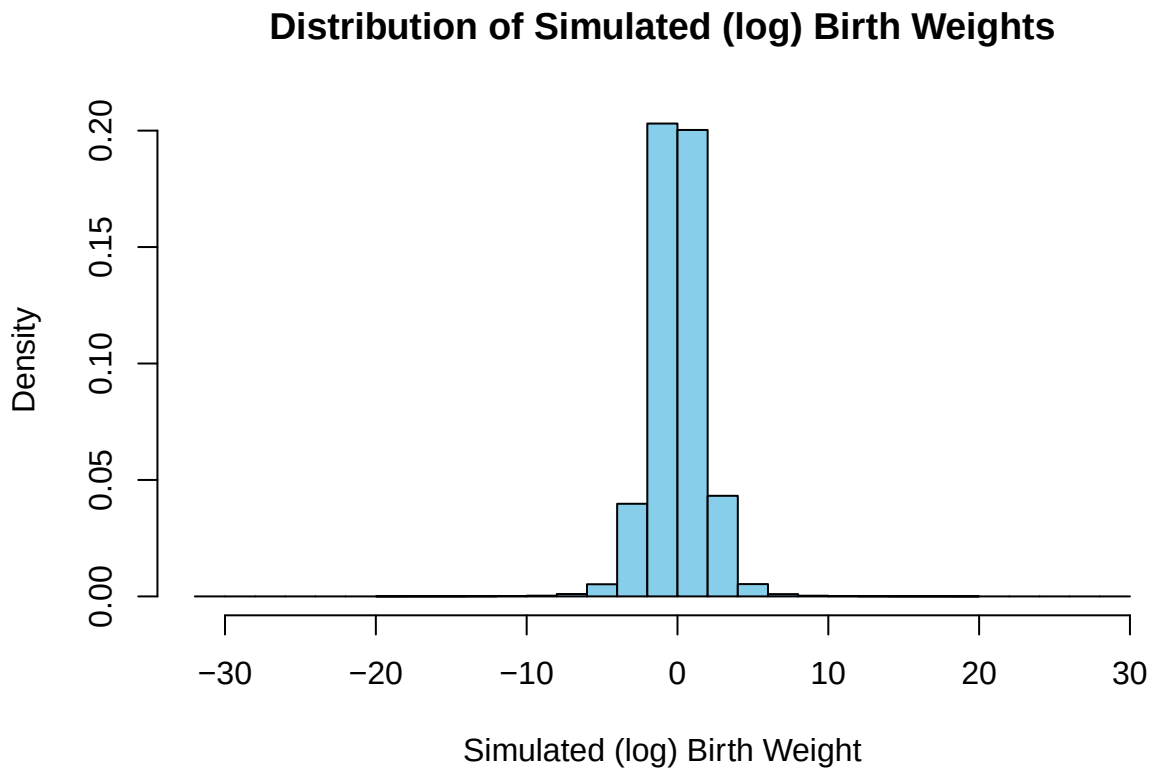
Simulating log birth weights:

```
set.seed(123)

sim_log_birthweights <- matrix(NA, nrow = N, ncol = nrow(ds))

for (i in 1:N) {
```

```
  sim_log_birthweights[i, ] <- rnorm(nrow(ds), mean = beta1[i] + beta2[i] * ds$log_gest_c, sd = sigma[i]
}
```

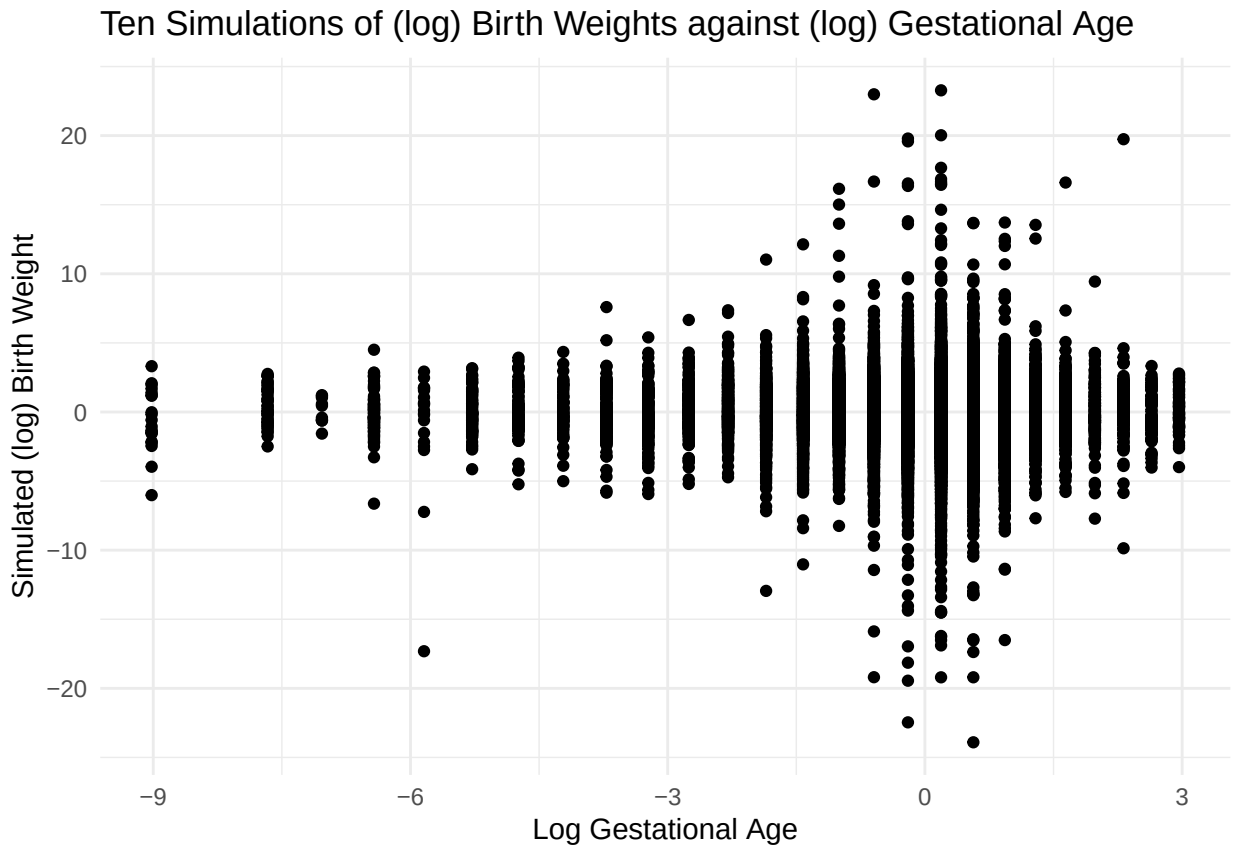
Plot the distribution of the simulated log birth weights:

```
all_sim <- as.vector(sim_log_birthweights)

hist(all_sim, main = "Distribution of Simulated (log) Birth Weights",
     xlab = "Simulated (log) Birth Weight", col = "skyblue", border = "black", probability = TRUE)
```




Plot ten simulations of (log) birth weights against (log) gestational age:

```
replicated_log_gest_c <- replicate(10, ds$log_gest_c)

plot_data <- data.frame(Log_Gestational_Age = as.vector(replicated_log_gest_c),
                        Simulated_Log_Birth_Weight = c(sim_log_birthweights[1:10, ]))

ggplot(plot_data, aes(x = Log_Gestational_Age, y = Simulated_Log_Birth_Weight)) +
  geom_point() +
  labs(title = "Ten Simulations of (log) Birth Weights against (log) Gestational Age",
       x = "Log Gestational Age",
       y = "Simulated (log) Birth Weight") +
  theme_minimal()
```




## Q3

Based on Model 1, give an estimate of the expected birthweight of a baby who was born at a gestational age of 37 weeks.

```
ds$log_weight <- log(ds$birthweight)

# put into a list
stan_data <- list(N = nrow(ds),
                  log_weight = ds$log_weight,
                  log_gest = ds$log_gest_c)

mod1 <- stan(data = stan_data,
             file = "code/models/simple_weight.stan",
             iter = 500,
             seed = 243)
```

```
## 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/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 0.000129 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.29 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: 0.215 seconds (Warm-up)
## Chain 1:                0.168 seconds (Sampling)
## Chain 1:                0.383 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 7.8e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.78 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: 0.211 seconds (Warm-up)
## Chain 2:                0.2 seconds (Sampling)
## Chain 2:                0.411 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 7.2e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.72 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: 0.214 seconds (Warm-up)
## Chain 3:                0.185 seconds (Sampling)
## Chain 3:                0.399 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 7.4e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.74 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 4: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 4: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 4: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 4: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 4: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 4: Iteration: 251 / 500 [ 50%]  (Sampling)
## Chain 4: Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 4: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 4: Iteration: 400 / 500 [ 80%]  (Sampling)
```

```
## Chain 4: Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 4: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.209 seconds (Warm-up)
## Chain 4:                0.185 seconds (Sampling)
## Chain 4:                0.394 seconds (Total)
## Chain 4:
```

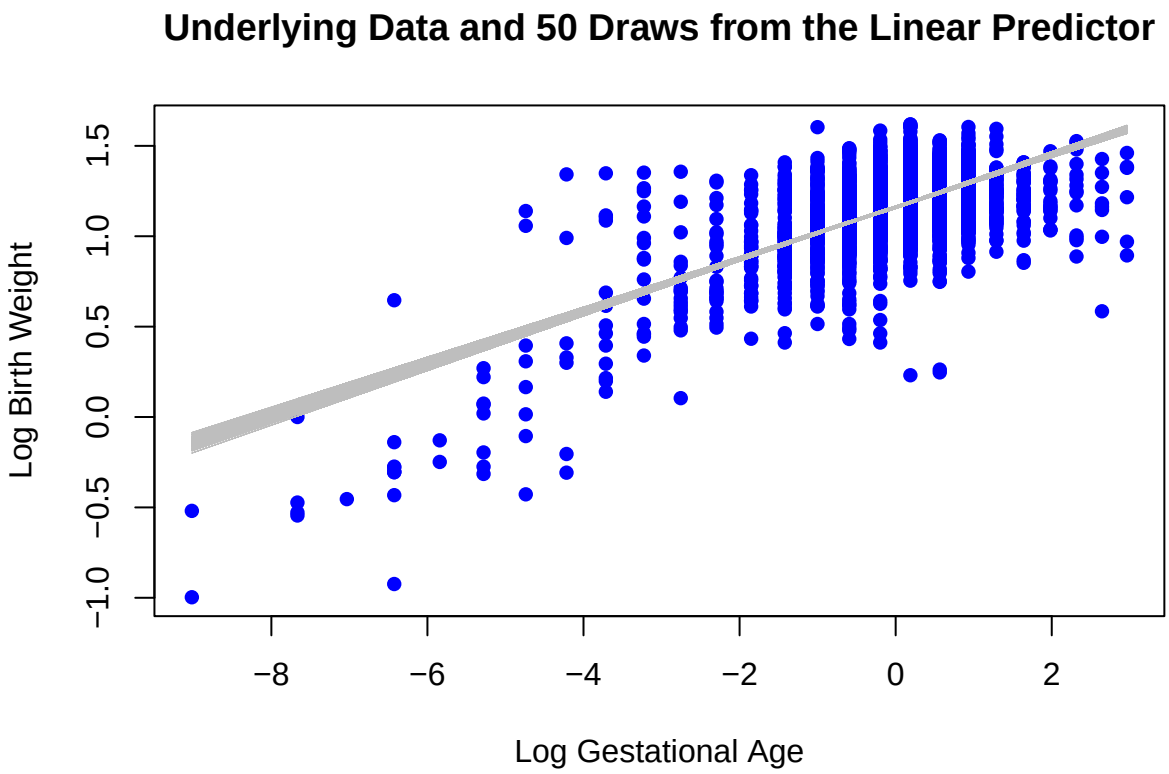
```r
gest <- 37
log_estimate <- summary(mod1)$summary["beta[1]","mean"] + summary(mod1)$summary["beta[2]","mean"] *
  (log(gest) - mean(log(ds$gest)))/sd(log(ds$gest))
estimate <- exp(log_estimate)
cat("The estimate of the expected birthweight of a baby who was born at a gestational age of 37 weeks:"
```

```
## The estimate of the expected birthweight of a baby who was born at a gestational age of 37 weeks: 2.9
```

## Q4

Based on Model 1, create a scatter plot showing the underlying data (on the appropriate scale) and 50 posterior draws of the linear predictor.

```r
# Extract posterior draws
set.seed(123)
posterior <- as.matrix(mod1)
draw_indices <- sample(1:dim(posterior)[1], size=50)
posterior_draws <- posterior[draw_indices, ]

# Extract relevant parameters
beta1_draws <- posterior_draws[, "beta[1]"]
beta2_draws <- posterior_draws[, "beta[2]"]
sigma_draws <- posterior_draws[, "sigma"]

# Create a matrix to store the linear predictor draws
linear_predictor_draws <- matrix(NA, nrow = 50, ncol = nrow(ds))

# Generate posterior draws of the linear predictors
for (i in 1:50) {
  linear_predictor_draws[i, ] <- beta1_draws[i] + beta2_draws[i] * ds$log_gest_c
}

# Plot the scatter plot with 50 posterior draws of the linear predictor
plot(ds$log_gest_c, ds$log_weight, pch = 16, col = "blue",
     xlab = "Log Gestational Age",
     ylab = "Log Birth Weight",
     main = "Underlying Data and 50 Draws from the Linear Predictor")

for (i in 1:50) {
  lines(ds$log_gest_c, linear_predictor_draws[i, ], col = "gray", lty = 1, lwd=0.1)
}
```




## Q5

Write a Stan model to run Model 2, and run it. Report a summary of the results, and interpret the coefficient estimate on the interaction term.

```
# encode preterm into numeric values
ds$preterm_enc <- ifelse(ds$preterm=="Y", 1, 0)

# put into a list
stan_data2 <- list(N = nrow(ds),
                   log_weight = ds$log_weight,
                   log_gest = ds$log_gest_c,
                   preterm = ds$preterm_enc)

mod2 <- stan(data = stan_data2,
             file = "code/models/model2.stan",
             iter = 500,
             seed = 243)
```

```
## 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/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/StanHeader
## 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 0.000371 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 3.71 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: 0.833 seconds (Warm-up)
## Chain 1:                0.654 seconds (Sampling)
## Chain 1:                1.487 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000187 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.87 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: 0.895 seconds (Warm-up)
## Chain 2:                0.823 seconds (Sampling)
## Chain 2:                1.718 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000193 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.93 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: 0.912 seconds (Warm-up)
## Chain 3:                0.652 seconds (Sampling)
## Chain 3:                1.564 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000188 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.88 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 4: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 4: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 4: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 4: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 4: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 4: Iteration: 251 / 500 [ 50%]  (Sampling)
## Chain 4: Iteration: 300 / 500 [ 60%]  (Sampling)
```

```
## Chain 4: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 4: Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 4: Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 4: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.934 seconds (Warm-up)
## Chain 4:                0.772 seconds (Sampling)
## Chain 4:                1.706 seconds (Total)
## Chain 4:
```

Print the result summary:

```
summary_mod2 <- summary(mod2)

# Extract estimates for beta1, beta2, beta3, beta4, and sigma
estimates <- summary_mod2$summary[c("beta[1]", "beta[2]", "beta[3]", "beta[4]", "sigma"), ]
print(estimates)
```

```
##              mean      se_mean          sd       2.5%       25%       50%
## beta[1] 1.1697984 6.988659e-05 0.002653556 1.16488936 1.1679383 1.1698020
## beta[2] 0.1020085 1.031041e-04 0.003530353 0.09531607 0.0995568 0.1018874
## beta[3] 0.5550383 3.332277e-03 0.066130187 0.43201346 0.5108913 0.5549890
## beta[4] 0.1969413 6.963686e-04 0.013430650 0.17169772 0.1876603 0.1972734
## sigma   0.1612443 8.144790e-05 0.001815965 0.15780069 0.1599945 0.1612466
##               75%     97.5%     n_eff      Rhat
## beta[1] 1.1717013 1.1749061 1441.6803 0.9968325
## beta[2] 0.1044938 0.1088995 1172.4229 0.9983309
## beta[3] 0.5977654 0.6834633  393.8377 1.0046655
## beta[4] 0.2063210 0.2238342  371.9767 1.0058074
## sigma   0.1625328 0.1647124  497.1130 1.0011154
```

The parameter for the interaction term is $\beta_4$ and the coefficient estimate is approximately 0.20. This means that for preterm births, a unit increase in the log of gestational age will lead to a 0.2 more unit of increase in the log of birth weight than non-preterm births.

## Q6

Make a similar plot to the one above but for Model 2, and not using the bayes plot in built function (i.e. do it yourself just with geom_density)

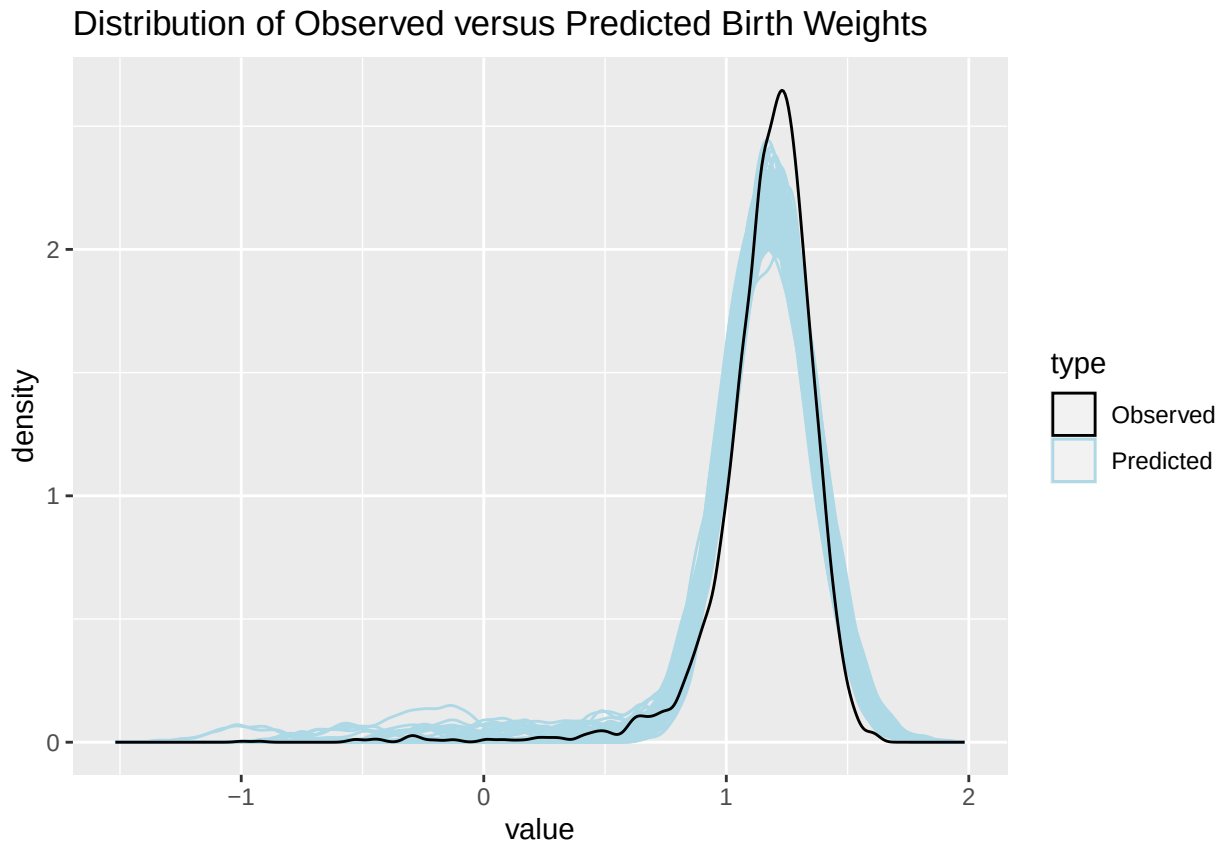
Drawing 100 datasets from the posterior predictive distribution of mod2:

```
set.seed(1856)
y <- ds$log_weight
yrep2 <- extract(mod2)[["log_weight_rep"]]

draw_indices <- sample(nrow(yrep2), size=100)
predicted_y <- yrep2[draw_indices, ]
dim(predicted_y)
```

```
## [1]  100 3842
```

Making the plot:

```
# Create a data frame for plotting
plot_data <- data.frame(
  value = c(as.vector(predicted_y), y),
  type = rep(c(rep("Predicted", nrow(predicted_y)), "Observed"), each = length(y)),
  draw = rep(0:nrow(predicted_y), each = length(y))
)

# Plot using geom_density
ggplot(plot_data, aes(x = value, color = type, group = interaction(type, draw))) +
  geom_density(alpha = 0) +
  scale_color_manual(values = c("black", "lightblue")) +
  scale_size_manual(values = c(10, 1)) +
  ggtitle("Distribution of Observed versus Predicted Birth Weights")
```




## Q7

Use a test statistic of the proportion of births under 2.5kg. Calculate the test statistic for the data, and the posterior predictive samples for both models, and plot the comparison (one plot per model).

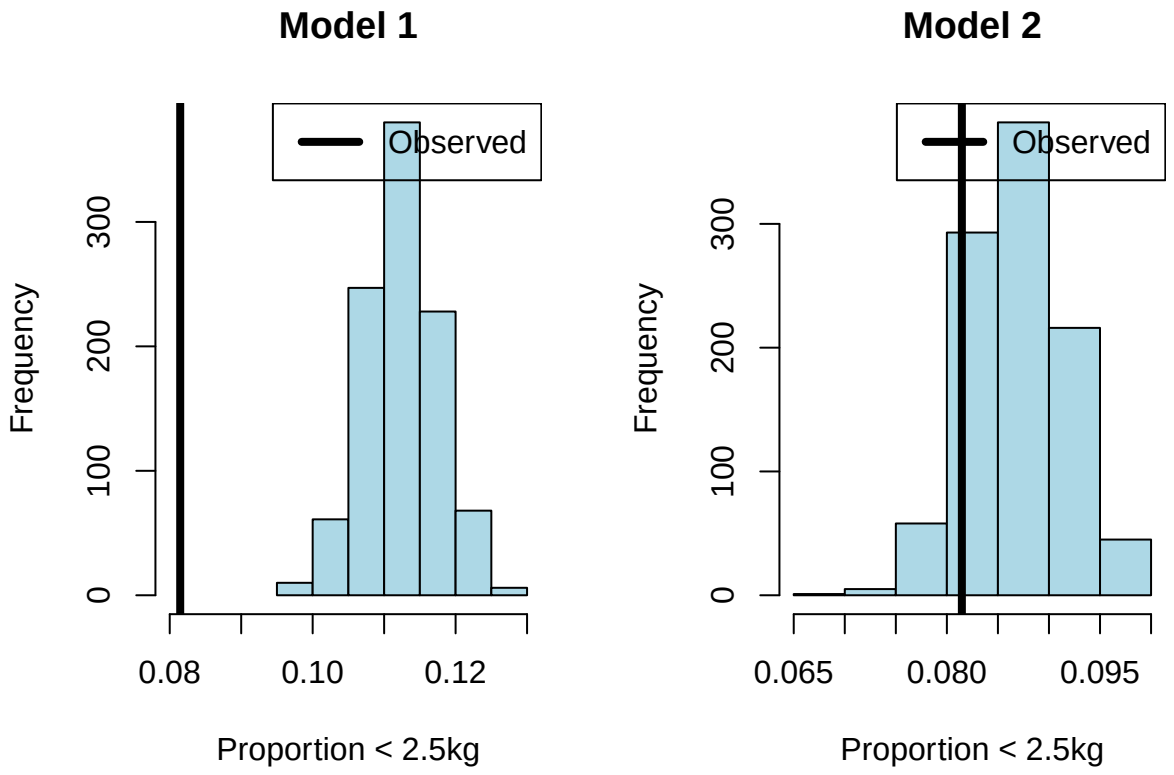
Calculating the observed and posterior test statistics:

```
observed_stats <- sum(ds$birthweight < 2.5) / nrow(ds)
```

```r
ppc_mod1 <- extract(mod1)[["log_weight_rep"]]
ppc_mod2 <- extract(mod2)[["log_weight_rep"]]

test_stat_mod1 <- rowMeans(ppc_mod1 < log(2.5))
test_stat_mod2 <- rowMeans(ppc_mod2 < log(2.5))
```

Plot the test statistics for model 1 and model 2:

```r
par(mfrow = c(1, 2))
hist(test_stat_mod1, main = "Model 1", xlab = "Proportion < 2.5kg", col = "lightblue", xlim = c(0.08, 0
abline(v = observed_stats, col = "black", lty = 1, lwd = 4)
legend("topright", legend = "Observed", col = "black", lty = 1, lwd = 4)

hist(test_stat_mod2, main = "Model 2", xlab = "Proportion < 2.5kg", col = "lightblue")
abline(v = observed_stats, col = "black", lty = 1, lwd = 4)
legend("topright", legend = "Observed", col = "black", lty = 1, lwd = 4)
```




## Q8

Get the LOO estimate of elpd for Model 2 and compare the two models with the loo_compare function. Interpret the results.

```r
loglik1 <- extract(mod1)[["log_lik"]]
loo1 <- loo(loglik1, save_psis = TRUE)

loglik2 <- extract(mod2)[["log_lik"]]
loo2 <- loo(loglik2, save_psis = TRUE)

compare_result <- loo_compare(loo1, loo2)
print(compare_result)
```

```
##        elpd_diff se_diff
## model2    0.0       0.0
## model1 -175.1      36.4
```

Based on the comparison result, we conclude that model 2 has a higher expected predictive performance (higher expected log predictive density) compared to model 1.

We can also calculate the 95% credible interval of the difference to see if it includes 0:

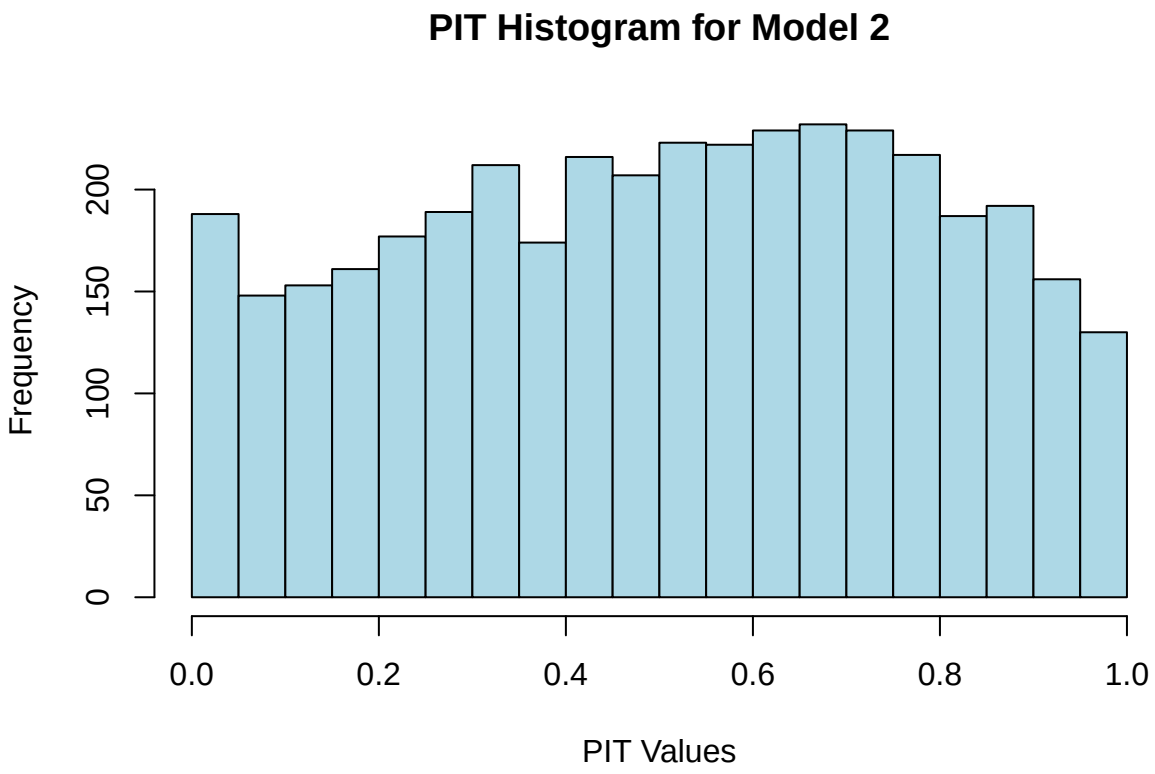
```r
credible_interval <- c(
  compare_result[2, "elpd_diff"] - 1.96 * compare_result[2, "se_diff"],
  compare_result[2, "elpd_diff"] + 1.96 * compare_result[2, "se_diff"]
)

print(credible_interval)
```

```
## [1] -246.4933 -103.7533
```

Since 0 is clearly not in the 95% credible interval, we can say that the difference in the expected predictive performance between model 1 and model 2 is statistically significant (model 2 is significantly better).

Create the PIT histogram for model 2 from scratch:

```r
observed_values <- ds$log_weight
pit_values_mod2 <- numeric(length(observed_values))

# Calculate PIT values:
posterior_mod2 <- extract(mod2)$log_weight_rep

for (i in seq_along(observed_values)) {
  pit_values_mod2[i] <- mean(posterior_mod2[, i] < observed_values[i])
}

hist(pit_values_mod2, breaks = 20, main = "PIT Histogram for Model 2", xlab = "PIT Values", col = "ligh
```




## Q9

Let's add mom's age to the model. First, we standardize the variable and take the log of it:

```
ds$log_mager_c <- (log(ds$mager) - mean(log(ds$mager)))/sd(log(ds$mager))
```

We add `log_bmi_c` to the new model 3:

```
# put into a list
stan_data3 <- list(N = nrow(ds),
                   log_weight = ds$log_weight,
                   log_gest = ds$log_gest_c,
                   preterm = ds$preterm_enc,
                   log_mager = ds$log_mager_c)

mod3 <- stan(data = stan_data3,
             file = "code/models/model3.stan",
             iter = 500,
             seed = 243)
```

```
## 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/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 0.000488 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 4.88 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: 0.911 seconds (Warm-up)
## Chain 1:                0.922 seconds (Sampling)
## Chain 1:                1.833 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000298 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.98 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: 1.041 seconds (Warm-up)
## Chain 2:                0.874 seconds (Sampling)
## Chain 2:                1.915 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000279 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.79 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: 1.068 seconds (Warm-up)
## Chain 3:                0.774 seconds (Sampling)
## Chain 3:                1.842 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000246 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.46 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 4: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 4: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 4: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 4: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 4: Iteration: 250 / 500 [ 50%]  (Warmup)
```

```
## Chain 4: Iteration: 251 / 500 [ 50%]  (Sampling)
## Chain 4: Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 4: Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 4: Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 4: Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 4: Iteration: 500 / 500 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 1.124 seconds (Warm-up)
## Chain 4:                0.748 seconds (Sampling)
## Chain 4:                1.872 seconds (Total)
## Chain 4:
```

Print the result summary:

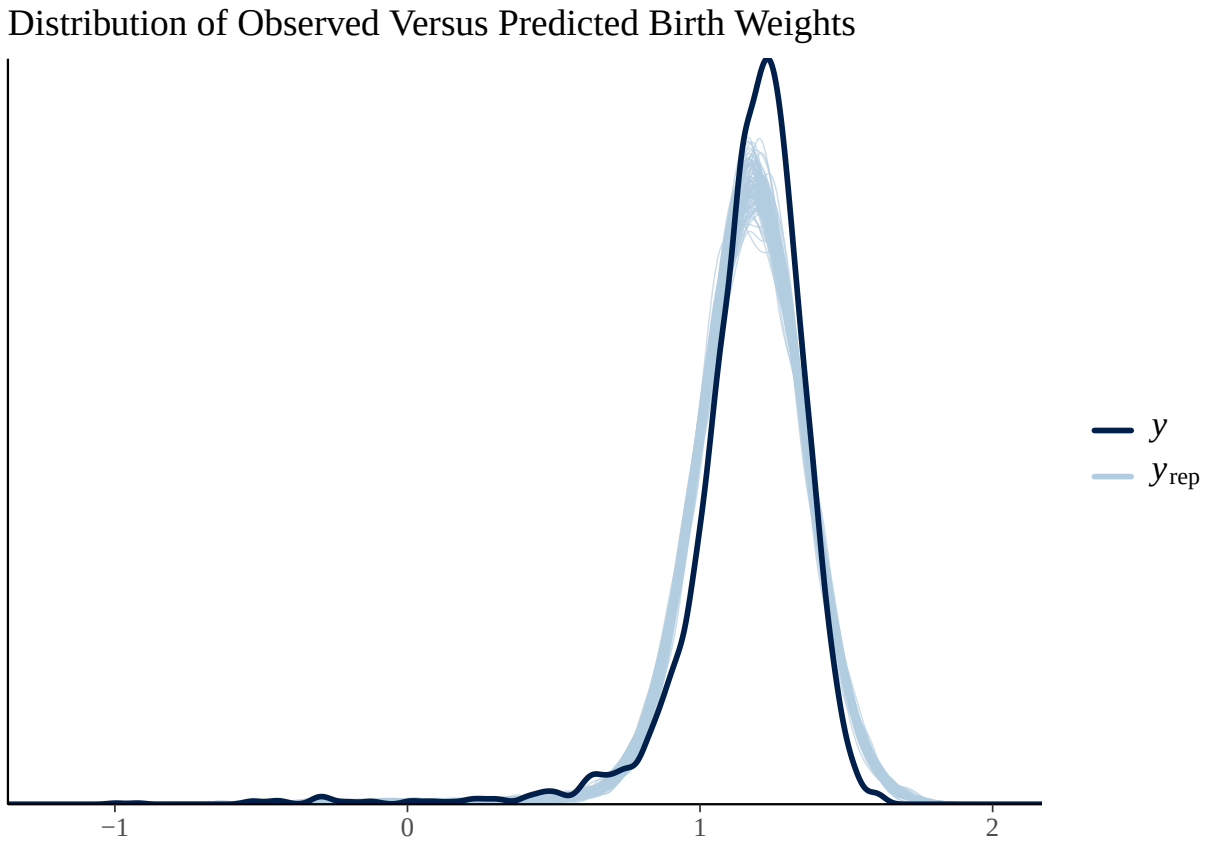
```
summary_mod3 <- summary(mod3)

# Extract estimates for beta1, beta2, beta3, beta4, and sigma
estimates <- summary_mod3$summary[c("beta[1]", "beta[2]", "beta[3]", "beta[4]", "beta[5]", "sigma"), ]
print(estimates)
```

```
##               mean      se_mean          sd        2.5%        25%        50%
## beta[1] 1.16965160 8.064724e-05 0.002568624 1.164545021 1.16797147 1.16974713
## beta[2] 0.10254897 1.209225e-04 0.003569428 0.095749180 0.10023118 0.10239110
## beta[3] 0.56273815 4.702402e-03 0.068160134 0.426710107 0.51951811 0.56671138
## beta[4] 0.19808076 9.496847e-04 0.014292592 0.167483986 0.18941323 0.19840540
## beta[5] 0.01522524 8.208532e-05 0.002672031 0.009881759 0.01355833 0.01530952
## sigma   0.16049167 9.200860e-05 0.001784188 0.157052162 0.15924705 0.16036165
##                75%      97.5%     n_eff      Rhat
## beta[1] 1.17130932 1.17453995 1014.4302 0.9976092
## beta[2] 0.10503248 0.10943502  871.3313 0.9996883
## beta[3] 0.60753421 0.69384165  210.0978 1.0072978
## beta[4] 0.20732796 0.22644244  226.4973 1.0055312
## beta[5] 0.01692198 0.02041018 1059.6247 0.9982942
## sigma   0.16170698 0.16416775  376.0317 1.0138349
```

The coefficient for $\beta_5$ can be interpreted as: on average, keep everything else the same, an unit increase in the log of mom's age, the log of the child's birth weight will increase by 0.015kg.

To perform PPC for model 3, we first plot 100 data sets from the posterior predictive distribution versus the observed data:

```
set.seed(123)
y <- ds$log_weight
yrep3 <- extract(mod3)[["log_weight_rep"]]

# draw 100 datasets from the posterior predictive distribution
samp100 <- sample(nrow(yrep3), 100)
ppc_dens_overlay(y, yrep3[samp100, ]) + ggtitle("Distribution of Observed Versus Predicted Birth Weights
```




By visually inspecting the plot, we don't see a big difference between this plot and the plot of model 2. So, we will calculate the LOO elpd for each model and compare.

```
loglik2 <- extract(mod2)[["log_lik"]]
loo2 <- loo(loglik2, save_psis = TRUE)

loglik3 <- extract(mod3)[["log_lik"]]
loo3 <- loo(loglik3, save_psis = TRUE)

compare_result <- loo_compare(loo2, loo3)
print(compare_result)
```

```
##        elpd_diff se_diff
## model2   0.0       0.0
## model1 -15.7       5.8
```

Based on the comparison result, we conclude that model 3 has a higher expected predictive performance (higher expected log predictive density) compared to model 2.

We can also calculate the 95% credible interval of the difference to see if it includes 0:

```
credible_interval <- c(
  compare_result[2, "elpd_diff"] - 1.96 * compare_result[2, "se_diff"],
  compare_result[2, "elpd_diff"] + 1.96 * compare_result[2, "se_diff"]
)

print(credible_interval)
```

```
## [1] -27.040068  -4.386059
```

Since 0 is not in the 95% credible interval, we can say that the difference in the expected predictive performance between model 2 and model 3 is statistically significant (model 3 is significantly better).