Week 6: Visualizing the Bayesian Workflow

13/03/24

Introduction

This lab will be looking at trying to replicate some of the visualizations in the lecture notes, involving prior and posterior predictive checks, and LOO model comparisons.

The dataset is a 0.1% of all births in the US in 2017. I've pulled out a few different variables, but as in the lecture, we'll just focus on birth weight and gestational age.

The data

Read it in, along with all our packages.

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

ds <- read_rds("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
```

Brief overview of variables:

- `mager` mum's age
- `mracehisp` mum's race/ethnicity see here for codes: https://data.nber.org/natality/2017/natl2017.pdf page 15
- `meduc` mum's education see here for codes: https://data.nber.org/natality/2017/natl2017.pdf page 16
- `bmi` mum's bmi
- `sex` baby's sex
- `combgest` gestational age in weeks
- `dbwt` birth weight in kg
- `ilive` alive at time of report y/n/ unsure

I'm going to rename some variables, remove any observations with missing gestational age or birth weight, restrict just to babies that were alive, and make a preterm variable.

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

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
- If you use `geom_smooth`, please also plot the underlying data

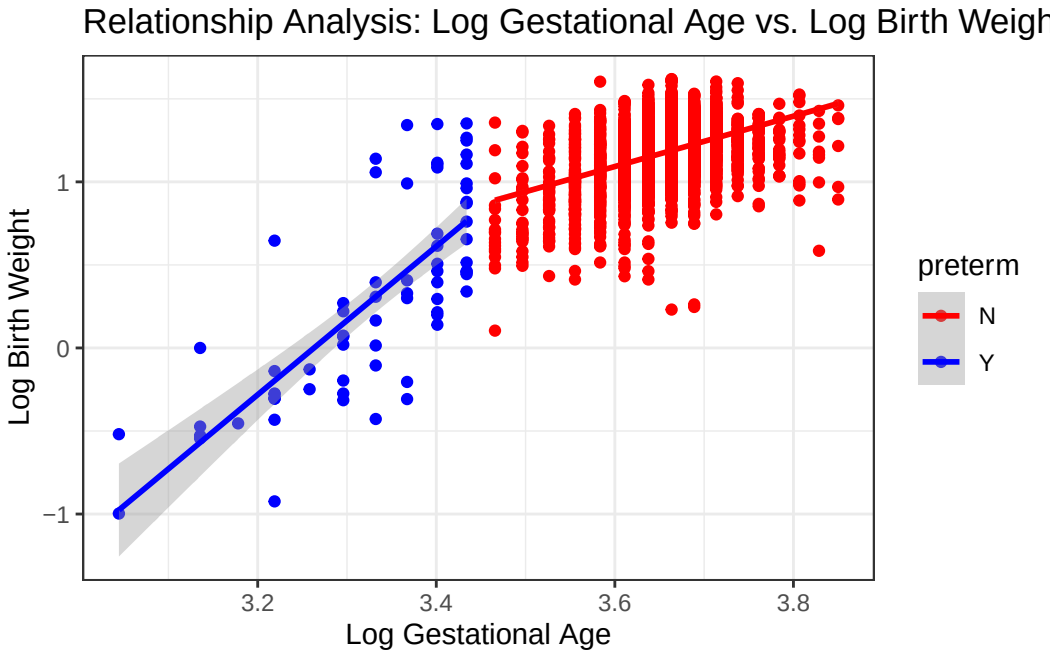
Feel free to replicate one of the scatter plots in the lectures as one of the interesting observations, as those form the basis of our models.

1 - Analysis of Birth Weight to Gestational Age

To begin, I used a scatter plot to analyze the relationship between the log of gestational age and the log of birth weight. We want to assess how gestational age correlates with birth weight and to observe any distinctions between preterm and full-term growth patterns. The results

shows that there is a positive correlation: as gestational age increases, birth weight increases as well, with a more pronounced growth rate during the preterm phase.

```
ggplot(ds, aes(x=log(gest), y = log(birthweight), color = preterm)) +
  geom_point() +
  geom_smooth(method = lm) +
  theme_bw() +
  scale_color_manual(values=c("red", "blue")) +
  labs(x = "Log Gestational Age", y="Log Birth Weight", title = "Relationship Analysis: Lo
```

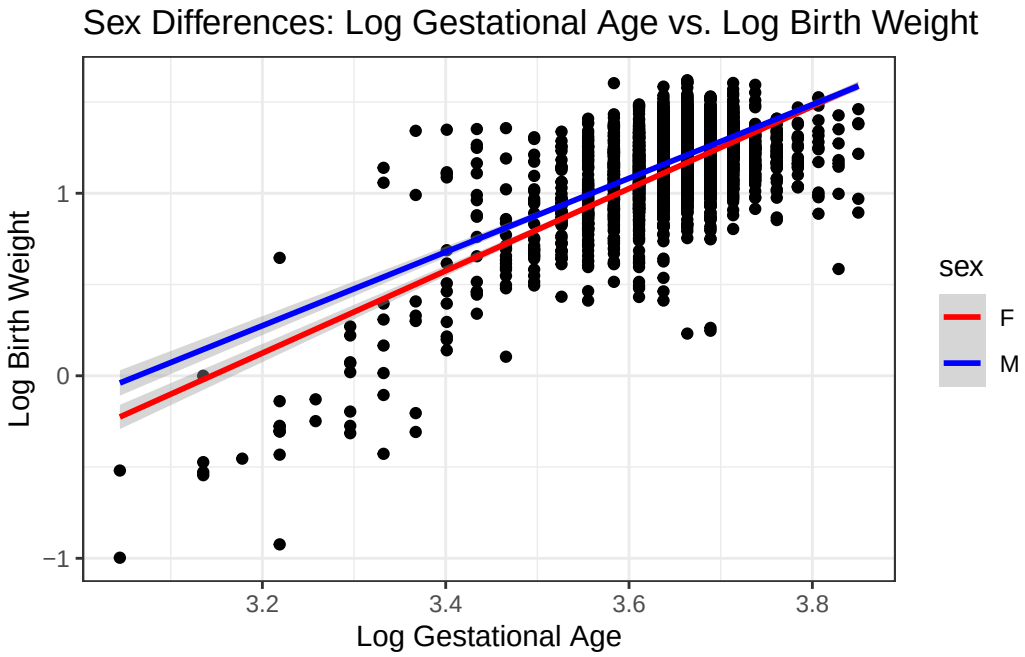


2 - Impact of Infant Sex on the Gestational Age-Birth Weight

This scatter plot, supplemented by a smoothed trend line for each sex, aims to uncover potential differences in the birth weight-gestational age relationship between male and female infants. We found that while birth weight generally increases with gestational age for both sexes, the growth trajectory is slightly steeper for girls compared to boys, though boys tend to weigh more at equivalent gestational ages.

```
ggplot(ds, aes(x=log(gest), y = log(birthweight))) +
  geom_point() +
  geom_smooth(method = lm,aes(color=sex)) +
  theme_bw() +
```

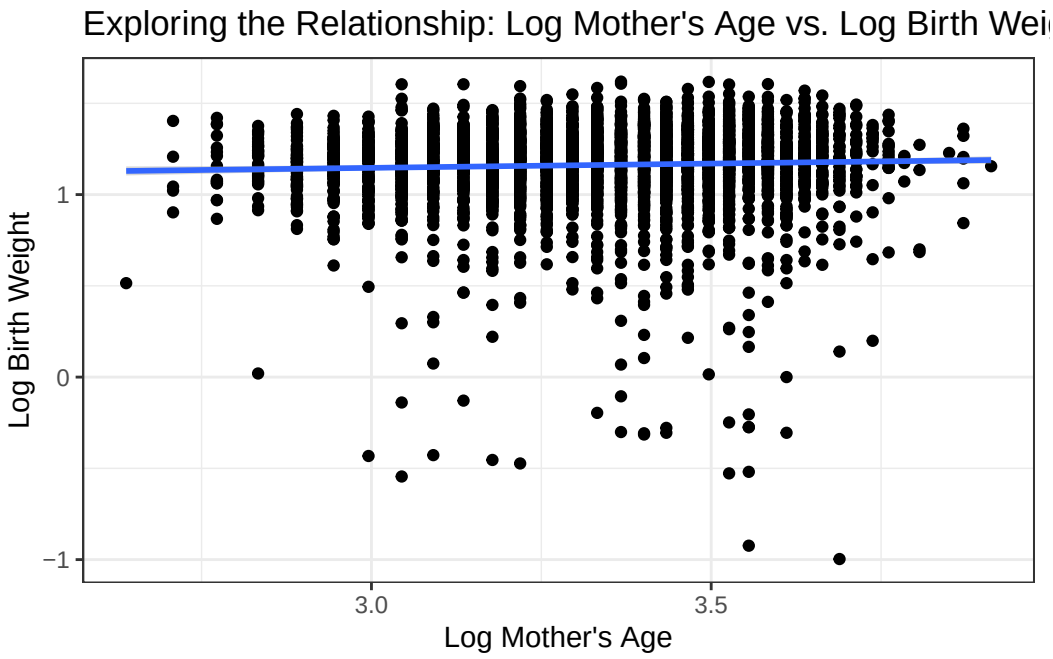
```
  scale_color_manual(values=c("red", "blue")) +
  labs(x = "Log Gestational Age", y="Log Birth Weight", title = "Sex Differences: Log Gest
```



3 - Investigating the Link Between Maternal Age and Infant Birth Weight

Through a scatter plot with a linear regression trend line, we want to determine the correlation between maternal age (logged) and infant birth weight (logged). We find that there is no significant trend of increasing or decreasing birth weight with maternal age. Nonetheless, a tendency toward lower birth weights at higher maternal ages is observed, potentially indicating a greater risk of preterm birth among older mothers.

```
ggplot(ds, aes(x=log(mager), y = log(birthweight))) +
  geom_point() +
  geom_smooth(method = lm) +
  theme_bw() +
  labs(x = "Log Mother's Age", y="Log Birth Weight", title = "Exploring the Relationship:
```



The model

As in lecture, we will look at two candidate models

Model 1 has log birth weight as a function of log gestational age

$$\log(y_i) \sim N(\beta_1 + \beta_2 \log(x_i), \sigma^2)$$

Model 2 has an interaction term between gestation and prematurity

$$\log(y_i) \sim N(\beta_1 + \beta_2 \log(x_i) + \beta_2 z_i + \beta_3 \log(x_i) z_i, \sigma^2)$$

- y_i is weight in kg
- x_i is gestational age in weeks, CENTERED AND STANDARDIZED
- z_i is preterm (0 or 1, if gestational age is less than 32 weeks)

Prior predictive checks

Let's put some weakly informative priors on all parameters i.e. for the βs

$$\beta \sim N(0, 1)$$

and for σ

$$\sigma \sim N^{+}(0, 1)$$

where the plus means positive values only i.e. Half Normal.

Let's check to see what the resulting distribution of birth weights look like given Model 1 and the priors specified above, assuming we had no data on birth weight (but observations of gestational age).

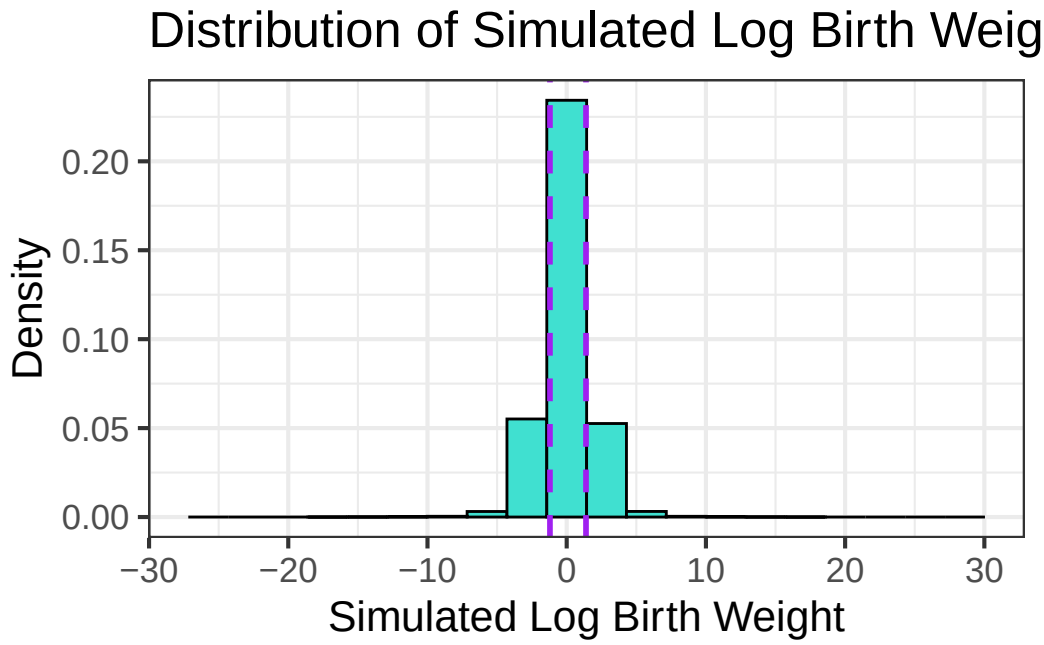
Question 2

For Model 1, simulate values of βs and σ based on the priors above. Do 1000 simulations. Use these values to simulate (log) birth weights from the likelihood specified in Model 1, based on the set of observed gestational weights. **Remember the gestational weights should be centered and standardized**.

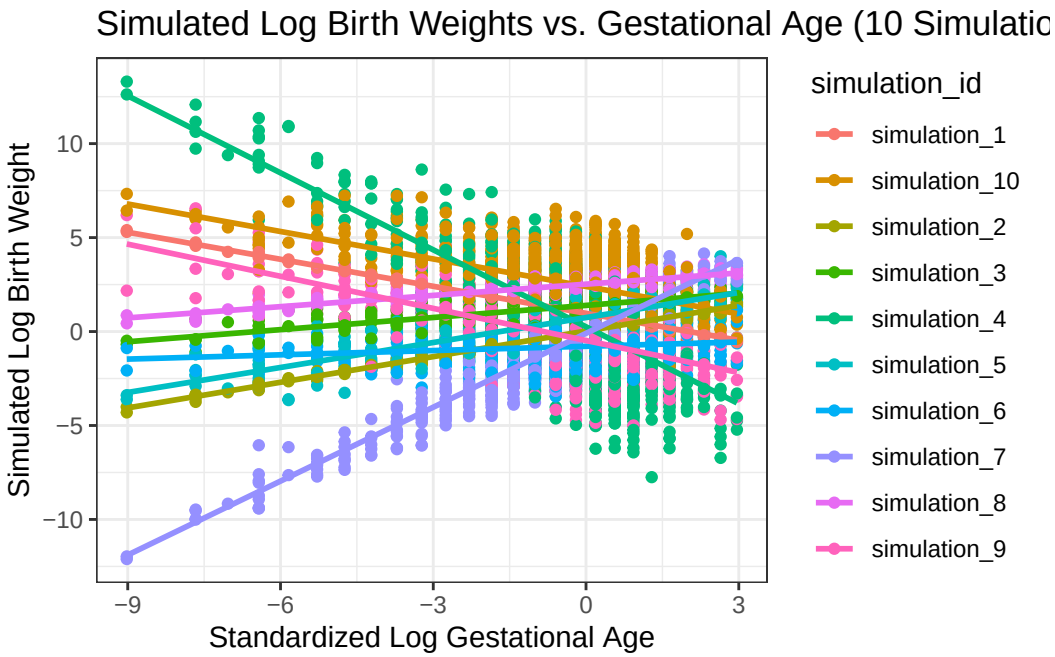
- Plot the resulting distribution of simulated (log) birth weights.
- Plot ten simulations of (log) birthweights against gestational age.

```
set.seed(2201)
num_simulations <- 1000

beta0_samples <- rnorm(num_simulations, mean = 0, sd = 1)
beta1_samples <- rnorm(num_simulations, mean = 0, sd = 1)
sigma_samples <- abs(rnorm(num_simulations, mean = 0, sd = 1))

ds$log_gestational_age_c <- scale(log(ds$gest))

# Simulate log birth weights for each set of parameters
simulated_data <- as_tibble(ds$log_gestational_age_c, .name_repair = "unique")
for (sim_index in 1:num_simulations) {
  simulated_mu <- beta0_samples[sim_index] + beta1_samples[sim_index] * ds$log_gestational
  column_name <- paste0("simulation_", sim_index)
  simulated_data[[column_name]] <- simulated_mu + rnorm(n = nrow(ds), mean = 0, sd = sigma
}
```

```r
# Plot the distribution of all simulated log birth weights
simulated_data_long <- pivot_longer(simulated_data, cols = starts_with("simulation"), name
ggplot(simulated_data_long, aes(x = simulated_log_birth_weight)) +
  geom_histogram(aes(y = ..density..), bins = 20, fill = "turquoise", color = "black") +
  theme_bw(base_size = 16) +
  geom_vline(xintercept = log(0.3), color = "purple", lwd = 1, lty = 2) +
  geom_vline(xintercept = log(4), color = "purple", lwd = 1, lty = 2) +
  labs(x = "Simulated Log Birth Weight", y = "Density", title = "Distribution of Simulated
```




```r
simulated_data$log_gestational_age_c <- ds$log_gestational_age_c

selected_simulations <- paste0("simulation_", 1:10)
simulated_data_longer <- pivot_longer(simulated_data, cols = selected_simulations, names_t

ggplot(simulated_data_longer, aes(x = log_gestational_age_c, y = simulated_log_birth_weigh
  geom_point() +
  geom_smooth(se = FALSE, method = "lm") +
  theme_bw() +
  labs(x = "Standardized Log Gestational Age", y = "Simulated Log Birth Weight", title = "
```



The distribution plot should show most simulated birth weights within the good range (log(0.3) to log(4) kg), marked by purple lines in the first figure. In the second plot, examining ten simulations against gestational age provides insights into the variability of the model's predictions. Ideally, all simulations should exhibit a positive correlation between gestational age and birth weight, consistent with expected biological relationships, but some simulation shows negative relationship between gestational age and birth body weight, which is contradicting to our EDA.

Run the model

Now we're going to run Model 1 in Stan. The stan code is in the `code/models` folder.

First, get our data into right form for input into stan.

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

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

Now fit the model

```
setwd("/Users/charlie/Desktop/STA2201/Lab 6")
```

```
mod1 <- stan(data = stan_data,
             file = "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 15.0.0 (clang-1500.3.9.4)'
using SDK: 'MacOSX14.4.sdk'
clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
#include <cmath>
         ^~~~~~~
1 error generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 9.5e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.95 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: 0.2 seconds (Warm-up)
Chain 1:                0.157 seconds (Sampling)
Chain 1:                0.357 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 5.9e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.59 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: 0.187 seconds (Warm-up)
Chain 2:                0.179 seconds (Sampling)
Chain 2:                0.366 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 6.4e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.64 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: 0.206 seconds (Warm-up)
Chain 3:                0.172 seconds (Sampling)
Chain 3:                0.378 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 6.9e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.69 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: 0.192 seconds (Warm-up)
Chain 4:                0.171 seconds (Sampling)
Chain 4:                0.363 seconds (Total)
Chain 4:
```

```
summary(mod1)$summary[c("beta[1]", "beta[2]", "sigma"),]
```

```
            mean      se_mean          sd      2.5%       25%       50%
```

```
beta[1] 1.1626293 8.109795e-05 0.002794886 1.1571530 1.1607401 1.1626100
beta[2] 0.1437074 7.050797e-05 0.002760482 0.1381359 0.1418908 0.1436313
sigma   0.1688448 1.025235e-04 0.001845326 0.1652251 0.1675565 0.1689364
             75%     97.5%    n_eff      Rhat
beta[1] 1.1646213 1.1679552 1187.705 0.9970491
beta[2] 0.1454944 0.1491751 1532.828 0.9972723
sigma   0.1700804 0.1722141  323.966 1.0095667
```

Question 3

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

```
new_age <- (log(37) - mean(log(ds$gest)))/sd(log(ds$gest))

beta0 <- summary(mod1)$summary[c("beta[1]", "beta[2]", "sigma"),][1,1]
beta1 <- summary(mod1)$summary[c("beta[1]", "beta[2]", "sigma"),][2,1]
birthweight <- exp(beta0 + beta1 * new_age)
birthweight
```

```
[1] 2.936397
```

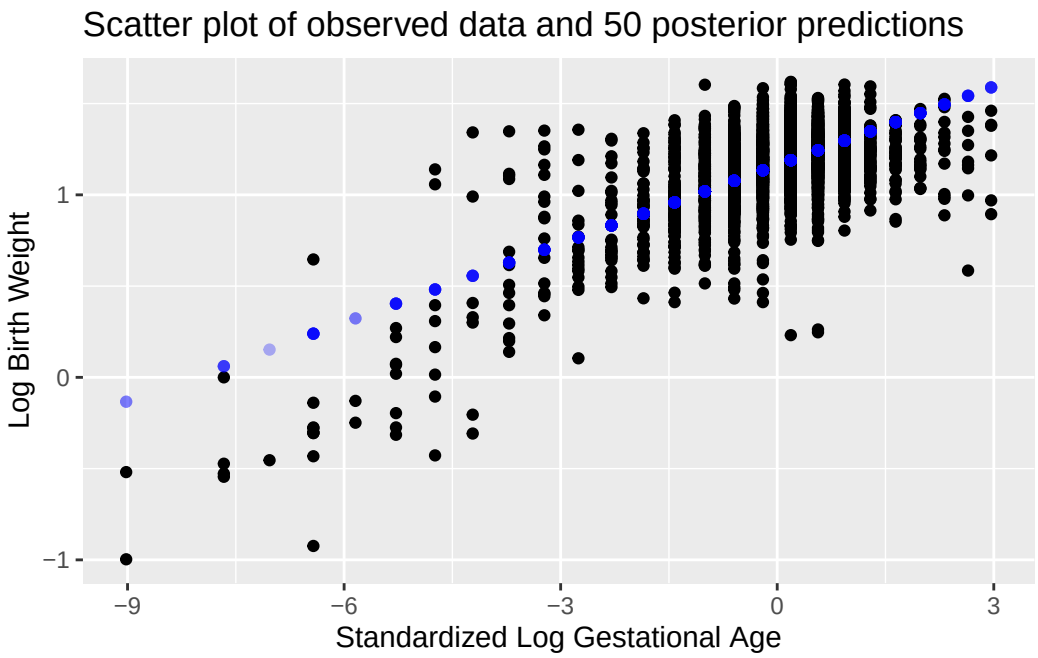
The expected birth weight of a baby who was born at a gestational age of 37 weeks is 2.936397 kg.

Question 4

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

```
pred_matrix <- matrix(ncol = length(ds$log_gest_c), nrow = 50)

for (i in 1:50) {
  pred_matrix[i,] <- beta0[i] + beta1[i] * ds$log_gest_c
}

draw_numbers <- rep(1:50, each = length(ds$log_gest_c))
predictions <- data.frame(
  draw = draw_numbers,
  gestation = rep(ds$log_gest_c, times = 50),
```

```
  log_weight_predicted = as.vector(t(pred_matrix))
)

observations <- data.frame(
  gestation = ds$log_gest_c,
  log_weight_observed = ds$log_weight
)

ggplot(data = observations, aes(x = gestation, y = log_weight_observed)) +
  geom_point(color = 'black') +
  geom_point(data = predictions, aes(x = gestation, y = log_weight_predicted), alpha = 0.3
  labs(title = "Scatter plot of observed data and 50 posterior predictions",
       x = "Standardized Log Gestational Age",
       y = "Log Birth Weight")
```



It shows that the predictive model did a good job in predicting the trend.

Question 5

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.

```r
ds$preterm_ind <- ifelse(ds$preterm == "Y", 1,0)

stan_model2 <- list(N = nrow(ds),
                          log_weight = ds$log_weight,
                          log_gest = ds$log_gest_c,
                          preterm = ds$preterm_ind,
                          intercept = ds$preterm_ind * ds$log_gest_c)
```

```r
mod2 <- stan(data = stan_model2,
             file = "simple_weight_2.stan",
             iter = 500,
             seed = 243)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: ‘Apple clang version 15.0.0 (clang-1500.3.9.4)’
using SDK: ‘MacOSX14.4.sdk’
clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
#include <cmath>
         ^~~~~~~
1 error generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000303 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 3.03 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: 0.644 seconds (Warm-up)
Chain 1:                0.506 seconds (Sampling)
Chain 1:                1.15 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000145 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.45 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: 0.682 seconds (Warm-up)
Chain 2:                0.629 seconds (Sampling)
Chain 2:                1.311 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000145 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.45 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: 0.699 seconds (Warm-up)
Chain 3:                0.502 seconds (Sampling)
Chain 3:                1.201 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000145 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.45 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: 0.714 seconds (Warm-up)
Chain 4:                0.593 seconds (Sampling)
Chain 4:                1.307 seconds (Total)
Chain 4:
```

```r
save(mod2, file = "mod2_my.Rdata")
```

```r
summary(mod2,pars=c("beta","sigma"))
```

```
$summary
             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

$c_summary
, , chains = chain:1

         stats
parameter      mean          sd       2.5%        25%       50%       75%
  beta[1] 1.1698114 0.002900399 1.16360768 1.16778416 1.1698756 1.1718209
  beta[2] 0.1021910 0.003531591 0.09519247 0.09965422 0.1021683 0.1046875
  beta[3] 0.5496205 0.057519866 0.44048177 0.50699136 0.5533368 0.5924013
  beta[4] 0.1957190 0.011687303 0.17227481 0.18761868 0.1958833 0.2032280
  sigma   0.1612136 0.001752024 0.15824602 0.15989474 0.1610281 0.1623970
         stats
parameter     97.5%
  beta[1] 1.1756063
  beta[2] 0.1089344
  beta[3] 0.6559134
  beta[4] 0.2186174
  sigma   0.1647275

, , chains = chain:2

         stats
parameter      mean          sd       2.5%        25%       50%       75%
  beta[1] 1.1698814 0.002557452 1.16530481 1.16815507 1.1699091 1.1715678
```

```
  beta[2] 0.1019120 0.003548955 0.09529355 0.09945529 0.1017506 0.1044016
  beta[3] 0.5594172 0.065798981 0.43827368 0.51315405 0.5573436 0.6029008
  beta[4] 0.1979240 0.014162529 0.17285219 0.18765678 0.1983401 0.2082118
  sigma   0.1611814 0.001879479 0.15772601 0.15977293 0.1613061 0.1625575
         stats
parameter     97.5%
  beta[1] 1.1751004
  beta[2] 0.1091119
  beta[3] 0.6834814
  beta[4] 0.2250143
  sigma   0.1646302

, , chains = chain:3

         stats
parameter      mean          sd       2.5%        25%       50%       75%
  beta[1] 1.1697396 0.002732738 1.16470201 1.16781599 1.1698677 1.1715916
  beta[2] 0.1021994 0.003689149 0.09493984 0.09977859 0.1019881 0.1049098
  beta[3] 0.5498037 0.072896122 0.41167924 0.50433810 0.5507870 0.5974868
  beta[4] 0.1955408 0.014377008 0.17012310 0.18467378 0.1963119 0.2052230
  sigma   0.1612644 0.001856594 0.15748430 0.16005242 0.1613398 0.1624820
         stats
parameter     97.5%
  beta[1] 1.1748789
  beta[2] 0.1093808
  beta[3] 0.6875172
  beta[4] 0.2225545
  sigma   0.1644234

, , chains = chain:4

         stats
parameter      mean          sd       2.5%        25%       50%       75%
  beta[1] 1.1697613 0.002411971 1.16557876 1.16806023 1.1695862 1.1716813
  beta[2] 0.1017316 0.003342209 0.09624996 0.09946758 0.1015300 0.1043053
  beta[3] 0.5613119 0.066916808 0.44389509 0.51561629 0.5604625 0.6002097
  beta[4] 0.1985816 0.013141298 0.17697332 0.19023884 0.1984404 0.2071416
  sigma   0.1613176 0.001780715 0.15822696 0.16009145 0.1612239 0.1625807
         stats
parameter     97.5%
  beta[1] 1.1739586
  beta[2] 0.1079577
  beta[3] 0.7019116
```

```
beta[4] 0.2255893
sigma   0.1648638
```

Now I assumed that there is a typo for the two beta 2 in the model 2, so I used beta 2 and beta 3 for each of them and the interaction term I termed beta 4. Now, from the summary table, the coefficient of the interaction term is 0.196, which means that the impact of gestational age on birth weight is stronger for preterm than for non-preterm by 0.196. This shows that if baby is preterm birth, one unit increase in log of gestational age, on average, will result in 0.196 units expected increase in log of birth weight.

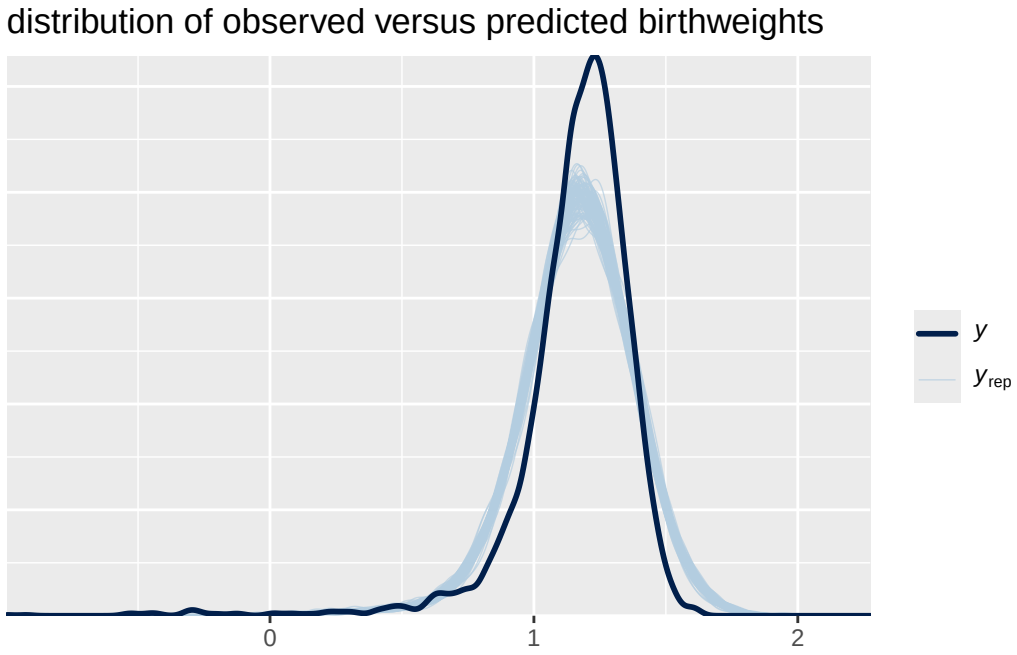
PPCs

Now we've run two candidate models let's do some posterior predictive checks. The `bayesplot` package has a lot of inbuilt graphing functions to do this. For example, let's plot the distribution of our data (y) against 100 different datasets drawn from the posterior predictive distribution:

```
set.seed(1856)
y <- ds$log_weight
yrep1 <- extract(mod1)[["log_weight_rep"]]
dim(yrep1)
```

```
[1] 1000 3842
```

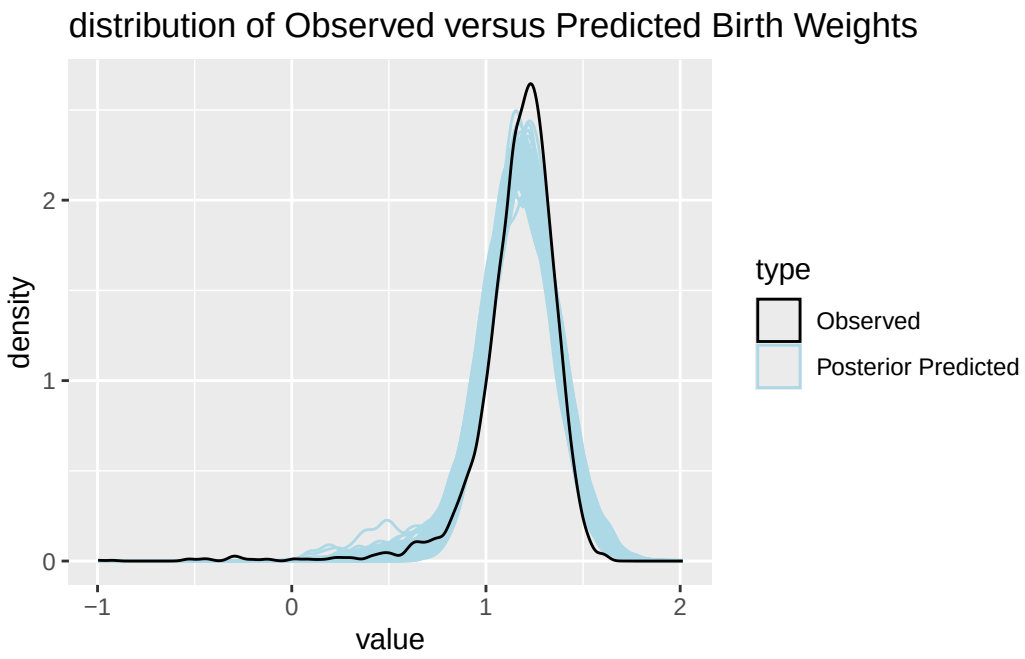
```
samp100 <- sample(nrow(yrep1), 100)
ppc_dens_overlay(y, yrep1[samp100, ])  + ggtitle("distribution of observed versus predicte
```



Question 6

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`)

```
fit2 <- extract(mod2)
yrep2 <- fit2$log_weight_rep
weight_post <- yrep2[samp100,]

df_plot <- data.frame(
  value = c(as.vector(weight_post), y),
  type = rep(c(rep("Posterior Predicted", nrow(weight_post)), "Observed"), each = length(y
  draws = rep(0:nrow(weight_post), each = length(y))
)

ggplot(df_plot, aes(x = value, color = type, group = interaction(type, draws))) +
  geom_density() +
  scale_color_manual(values = c("black", "lightblue")) +
  ggtitle("distribution of Observed versus Predicted Birth Weights")
```



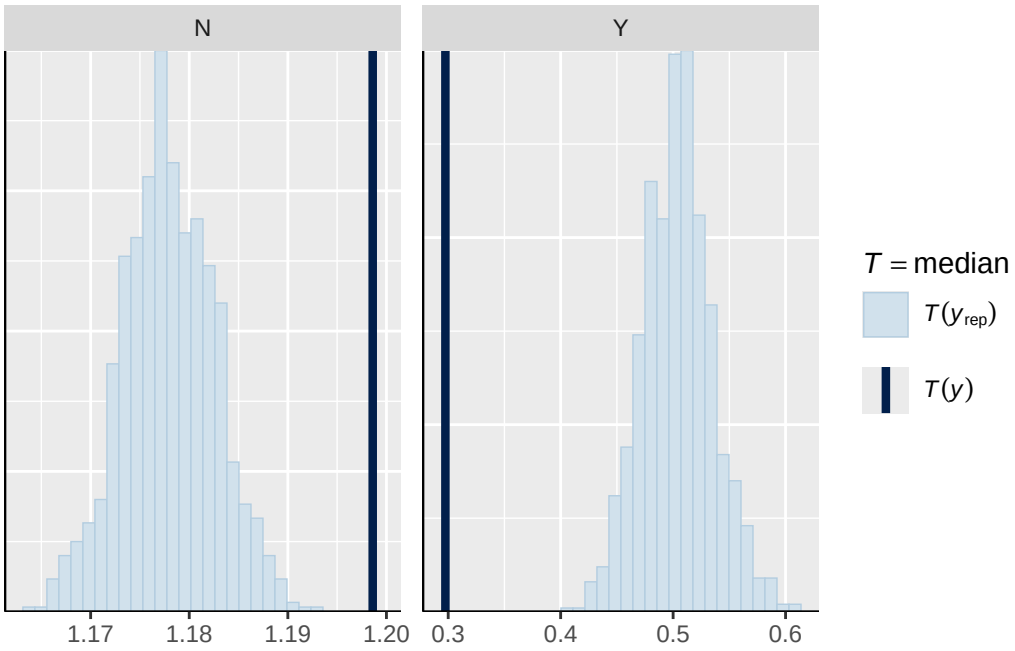
As it shows that model 2 has a much better posterior estmation than model 1, with the additional predictors.

Test statistics

We can also look at some summary statistics in the PPD versus the data, again either using `bayesplot` – the function of interest is `ppc_stat` or `ppc_stat_grouped` – or just doing it ourselves using ggplot.

E.g. medians by prematurity for Model 1

```
ppc_stat_grouped(ds$log_weight, yrep1, group = ds$preterm, stat = 'median')
```



Question 7

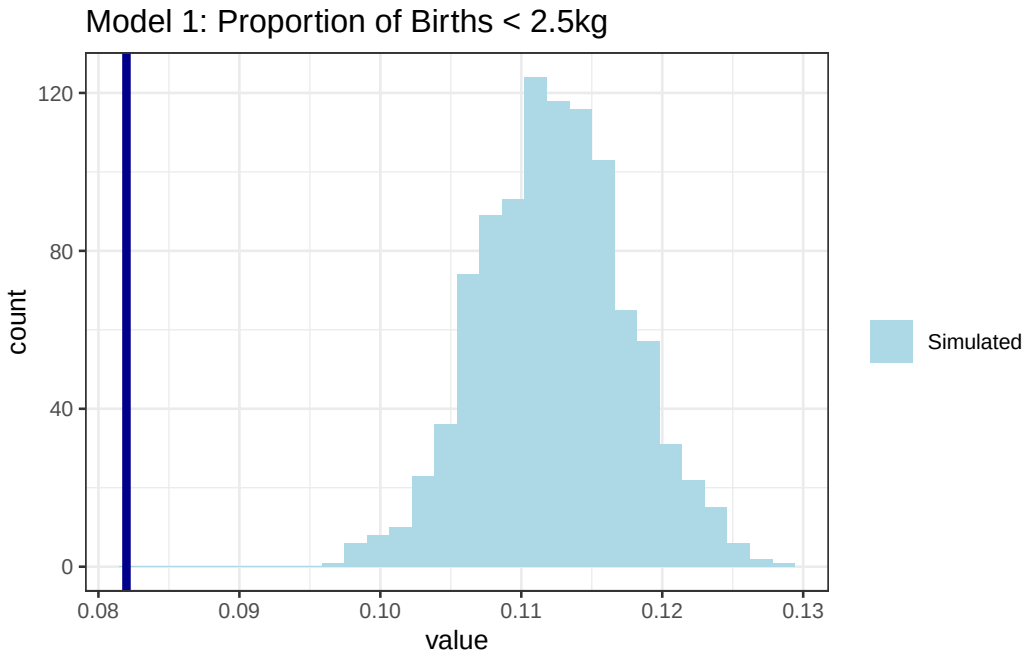
Use a test statistic of the proportion of births under 2.5kg. Calculate the test statistic for the data, and the posterior predictive samples for both models, and plot the comparison (one plot per model).

```
# Test statistic for the original data
original_log_weight <- ds$log_weight
proportion_under_2_5kg_original <- mean(original_log_weight <= log(2.5))
proportion_under_2_5kg_original
```

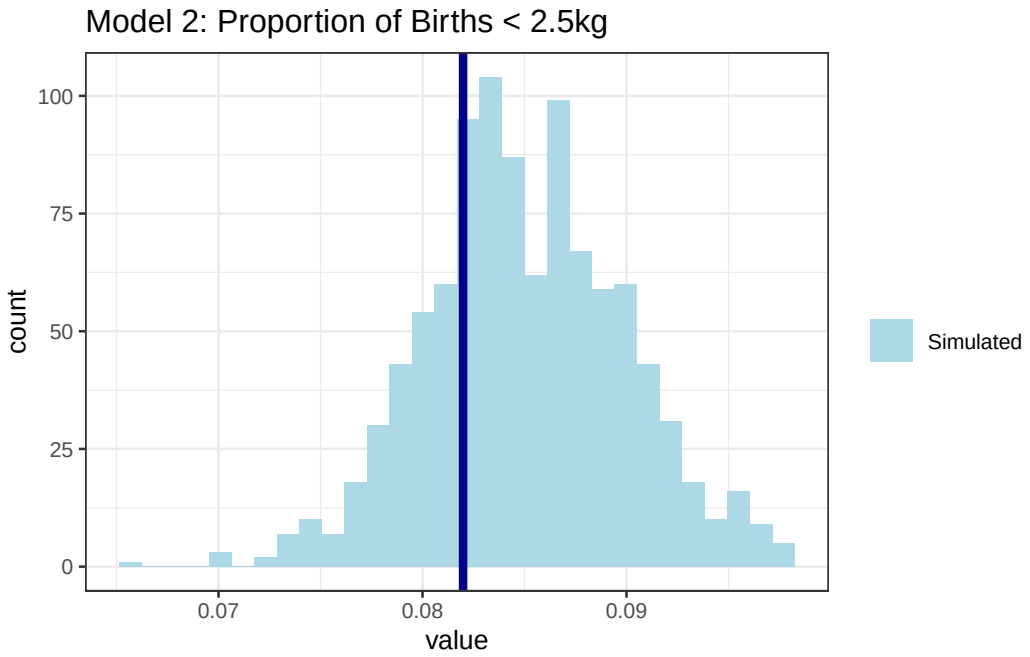
```
[1] 0.08198855
```

```
# Test statistics for simulations from Model 1
proportion_under_2_5kg_model1 <- sapply(1:nrow(yrep1), function(i) mean(yrep1[i,] <= log(2

# Test statistics for simulations from Model 2
proportion_under_2_5kg_model2 <- sapply(1:nrow(yrep2), function(i) mean(yrep2[i,] <= log(2

# Histogram for simulated data from Model 1 with observed test statistic line
ggplot(as_tibble(proportion_under_2_5kg_model1), aes(x = value)) +
```

```
  geom_histogram(aes(fill = "Simulated"), bins = 30, boundary = 0) +
  geom_vline(xintercept = proportion_under_2_5kg_original, color = "darkblue", lwd = 1.5)
  ggtitle("Model 1: Proportion of Births < 2.5kg") +
  theme_bw(base_size = 10) +
  scale_fill_manual(name = "", values = c("Simulated" = "lightblue"))
```




```
# Histogram for simulated data from Model 2 with observed test statistic line
ggplot(as_tibble(proportion_under_2_5kg_model2), aes(x = value)) +
  geom_histogram(aes(fill = "Simulated"), bins = 30, boundary = 0) +
  geom_vline(xintercept = proportion_under_2_5kg_original, color = "darkblue", lwd = 1.5)
  ggtitle("Model 2: Proportion of Births < 2.5kg") +
  theme_bw(base_size = 10) +
  scale_fill_manual(name = "", values = c("Simulated" = "lightblue"))
```



Observation:

The calculated test statistic for the original data indicates that approximately 8.2% of newborns weigh less than 2.5kg. The comparative histograms for both models show the distribution of this statistic across simulations. From these visualizations, it is apparent that the observed proportion aligns more closely with the predictions from Model 2. The histogram for Model 1 suggests a higher predicted proportion of newborns under the 2.5kg threshold, deviates from the observed data.

LOO

Finally let's calculate the LOO elpd for each model and compare. The first step of this is to get the point-wise log likelihood estimates from each model:

```
loglik1 <- extract(mod1)[["log_lik"]]
```

And then we can use these in the `loo` function to get estimates for the elpd. Note the `save_psis = TRUE` argument saves the calculation for each simulated draw, which is needed for the LOO-PIT calculation below.

```
loo1 <- loo(loglik1, save_psis = TRUE)
```

Look at the output:

```
loo1
```

```
Computed from 1000 by 3842 log-likelihood matrix.

         Estimate    SE
elpd_loo   1377.4  72.6
p_loo         9.3   1.4
looic     -2754.8 145.2
------
MCSE of elpd_loo is 0.1.
MCSE and ESS estimates assume independent draws (r_eff=1).

All Pareto k estimates are good (k < 0.67).
See help('pareto-k-diagnostic') for details.
```

Question 8

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

```
loglik2 <- fit2$log_lik
loo2 <- loo(loglik2, save_psis = TRUE)
loo2
```

```
Computed from 1000 by 3842 log-likelihood matrix.

         Estimate    SE
elpd_loo   1227.0 105.7
p_loo        26.3   4.9
looic     -2454.0 211.4
------
MCSE of elpd_loo is NA.
MCSE and ESS estimates assume independent draws (r_eff=1).

Pareto k diagnostic values:
                         Count Pct.    Min. ESS
(-Inf, 0.67]   (good)     3840  99.9%   263
```

```
   (0.67, 1]   (bad)         2   0.1%   <NA>
    (1, Inf)   (very bad)    0   0.0%   <NA>
See help('pareto-k-diagnostic') for details.
```

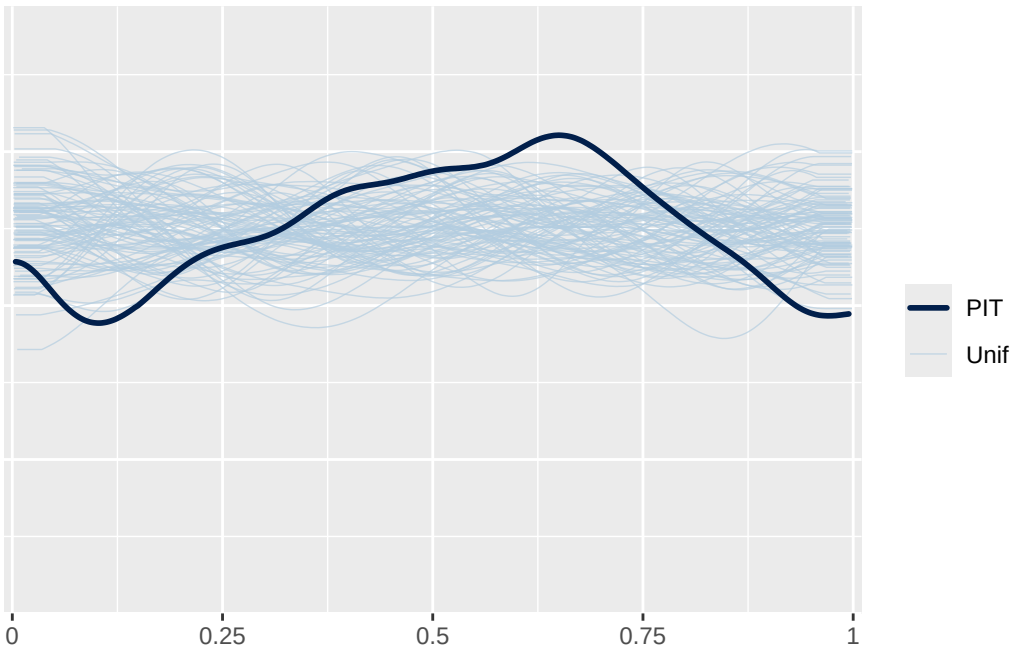
```
loo_comparison <- loo_compare(loo1, loo2)
loo_comparison
```

```
       elpd_diff se_diff
model1    0.0       0.0
model2 -150.4      50.8
```

The above suggests that model 1 might be a better fit than model 2. However, it is worth noting that some pareto k values are too high and thus the result might not be accurate for model 2.

We can also compare the LOO-PIT of each of the models to standard uniforms. For example for Model 1:

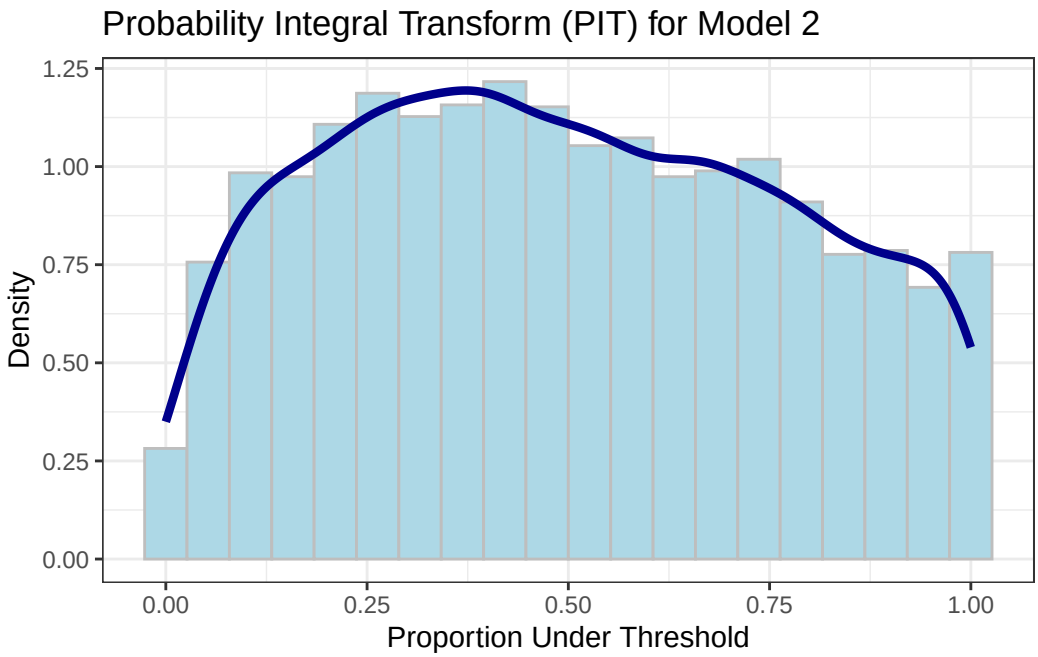
```
ppc_loo_pit_overlay(yrep = yrep1, y = y, lw = weights(loo1$psis_object))
```



Bonus question (not required)

Create your own PIT histogram "from scratch" for Model 2.

```
# Prepare the simulated data from Model 2 for analysis
set.seed(3729)
simulated_log_weights <- extract(mod2)[["log_weight_rep"]]
selected_samples <- sample(nrow(simulated_log_weights), 100)

# Convert the simulated weights to a tibble and add observed weights
simulated_data <- as_tibble(t(simulated_log_weights)) %>%
  bind_cols(observation_index = 1:nrow(ds), observed_log_weight = log(ds$birthweight))

# Reshape the data into a long format suitable for ggplot
long_simulated_data <- simulated_data %>%
  pivot_longer(cols = -c(observation_index, observed_log_weight), names_to = "simulation",

# Add a column to indicate if the observed weight is less than or equal to the simulated w
long_simulated_data <- long_simulated_data %>%
  mutate(under_threshold = as.numeric(observed_log_weight <= simulated_log_weight))

# Group by observation index and calculate the proportion for the Probability Integral Tra
pit_summary <- long_simulated_data %>%
  group_by(observation_index) %>%
  summarize(proportion_under_threshold = mean(under_threshold))

# Generate a histogram with a density plot for the PIT
pit_summary %>%
  ggplot(aes(x = proportion_under_threshold)) +
  geom_histogram(aes(y = ..density..), bins = 20, fill = "lightblue", color = "grey") +
  geom_density(color = "darkblue", lwd = 1.5) +
  labs(x = "Proportion Under Threshold", y = "Density",
       title = "Probability Integral Transform (PIT) for Model 2") +
  theme_bw()
```



Question 9

Based on the original dataset, choose one (or more) additional covariates to add to the linear regression model. Run the model in Stan, and compare with Model 2 above on at least 2 posterior predictive checks.

I brought the variable "sex" into the model as an extra feature.

$$\log(y_i) \sim N(\beta_1 + \beta_2 \log(x_i) + \beta_3 z_i + \beta_4 \log(x_i) z_i + \beta_5 s_i, \sigma^2)$$

where

- y_i is weight in kg
- x_i is gestational age in weeks
- z_i is preterm (0 or 1, if gestational age is less than 32 weeks)
- s_i is sex (0 or 1, for female or male)

```
# indicator for sex
ds$sex_ind <- ifelse(ds$preterm == "M", 1,0)

# prepare data
stan_data_model3 <- list(N = nrow(ds),
                         log_weight = ds$log_weight,
```

```
                        log_gest = ds$log_gest_c,
                        preterm = ds$preterm_ind,
                        intercept = ds$preterm_ind * ds$log_gest_c,
                        sex = ds$sex_ind)
```

```
mod3 <- stan(data = stan_data_model3,
             file = "simple_weight_3.stan",
             iter = 500,
             seed = 243)
```

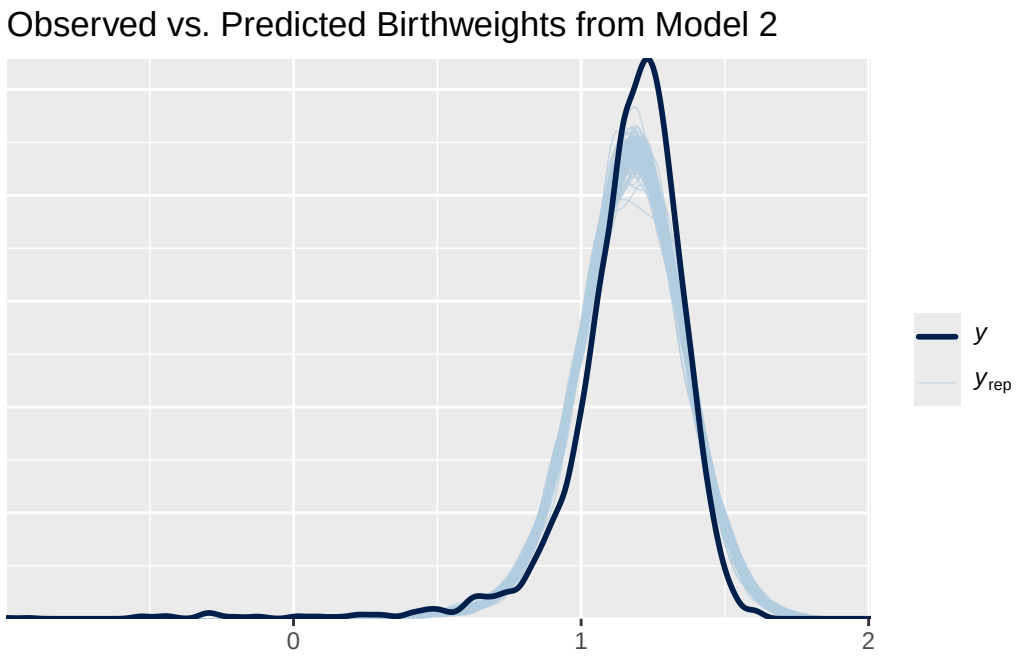
```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: 'Apple clang version 15.0.0 (clang-1500.3.9.4)'
using SDK: 'MacOSX14.4.sdk'
clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
#include <cmath>
         ^~~~~~~
1 error generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000391 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 3.91 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: 2.09 seconds (Warm-up)
Chain 1:                0.671 seconds (Sampling)
Chain 1:                2.761 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000189 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.89 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: 3.093 seconds (Warm-up)
Chain 2:                0.78 seconds (Sampling)
Chain 2:                3.873 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000196 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.96 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)
```

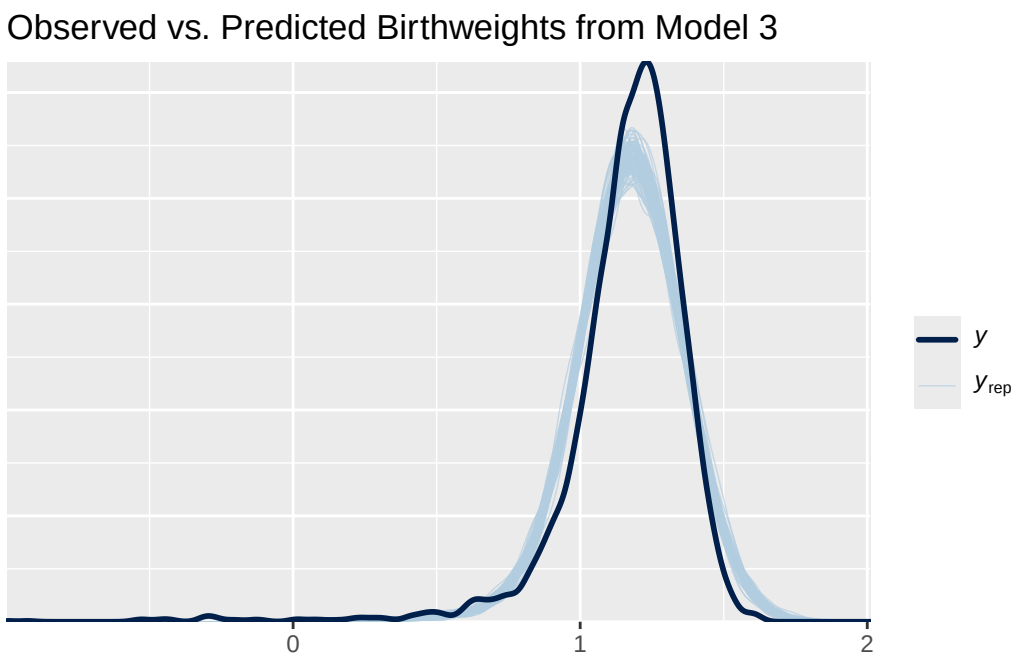
```
Chain 3: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 3: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 3: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 3: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 3: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 3: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 3: 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: 3.399 seconds (Warm-up)
Chain 3:                0.728 seconds (Sampling)
Chain 3:                4.127 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000202 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.02 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: 2.536 seconds (Warm-up)
Chain 4:                0.774 seconds (Sampling)
Chain 4:                3.31 seconds (Total)
Chain 4:
```

1. To assess model performance, we look at the posterior predictive distributions for model 2 and 3:

```r
set.seed(2201)

# Extract posterior predictive data
yrep_mod2 <- extract(mod2)[["log_weight_rep"]]
yrep_mod3 <- extract(mod3)[["log_weight_rep"]]

# Sample 100 simulated datasets for comparison
sample_indices <- sample(nrow(yrep_mod2), 100)

# Compare observed data to simulations for Model 2 and Model 3
ppc_dens_overlay(y, yrep_mod2[sample_indices, ]) +
  ggtitle("Observed vs. Predicted Birthweights from Model 2")
```




```r
ppc_dens_overlay(y, yrep_mod3[sample_indices, ]) +
  ggtitle("Observed vs. Predicted Birthweights from Model 3")
```

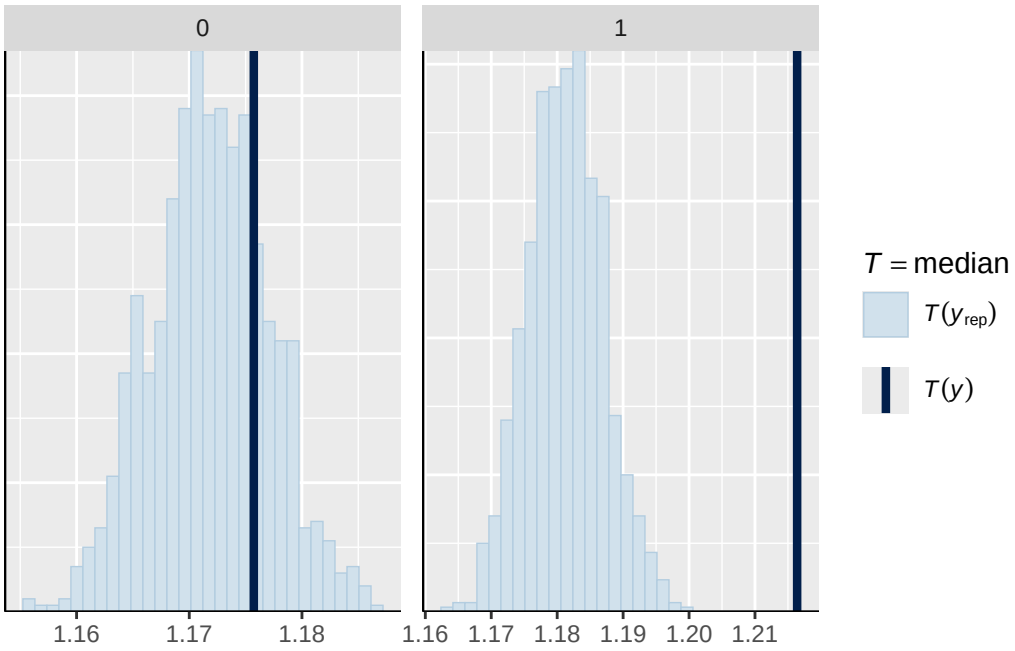


We see that both model looks similar and do not deviate much from the observed data.

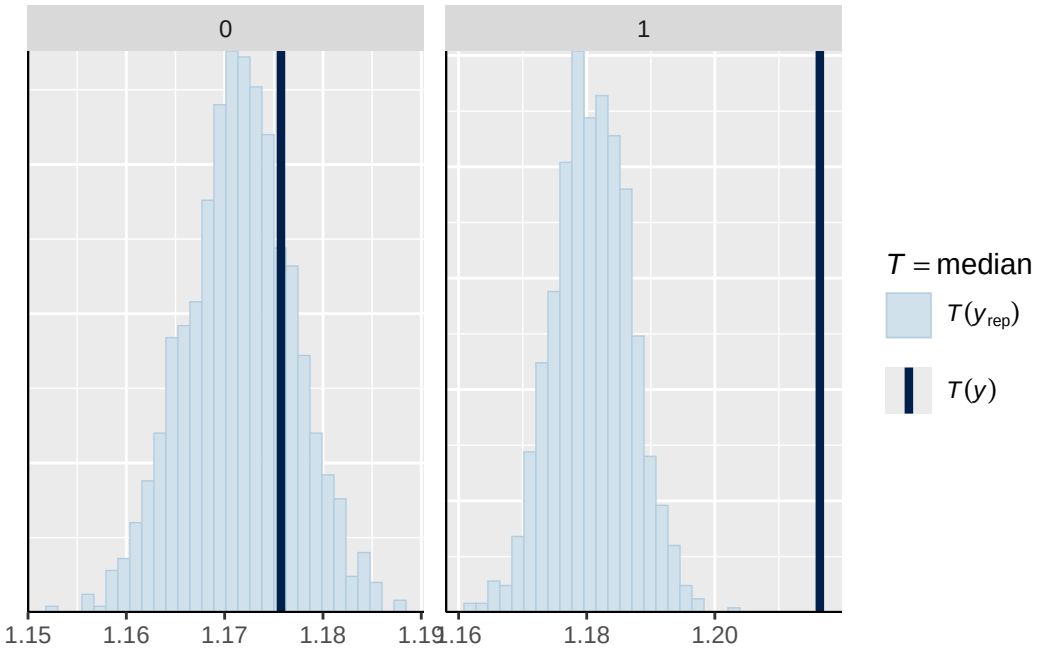
2. Test Statistics: median of different education between the two model

We further examine the median birthweight for different education levels by constructing a binary indicator for higher education based on the "meduc" variable:

```
# Binary indicator for higher education level
ds$higher_edu <- ifelse(ds$meduc > 4, 1, 0)

# Posterior predictive checks by education level
ppc_stat_grouped(ds$log_weight, yrep_mod2, group = ds$higher_edu, stat = 'median')
```



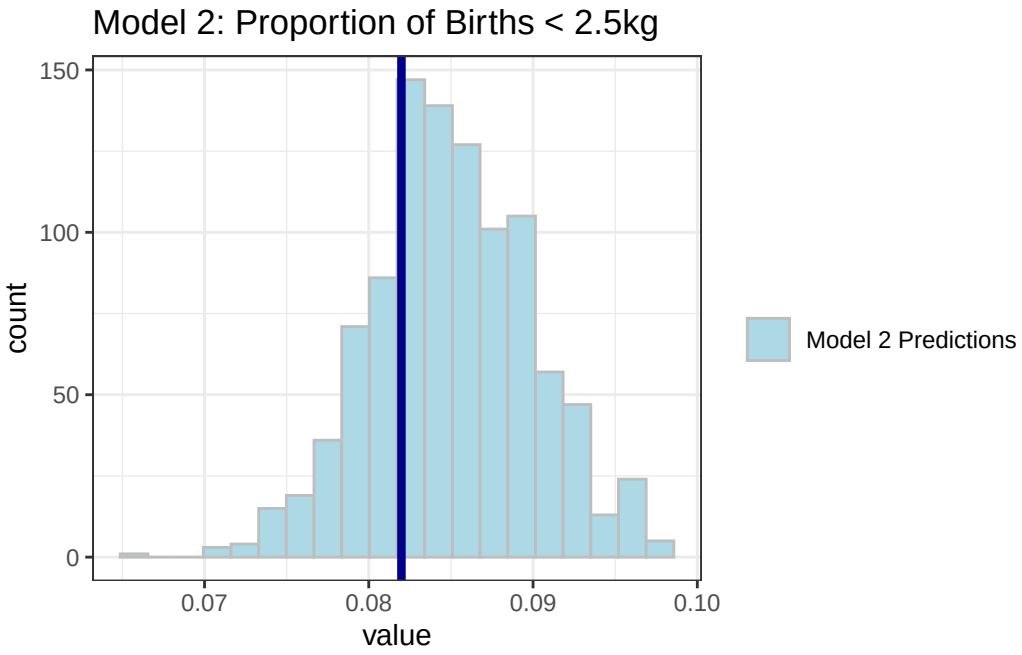

```
ppc_stat_grouped(ds$log_weight, yrep_mod3, group = ds$higher_edu, stat = 'median')
```



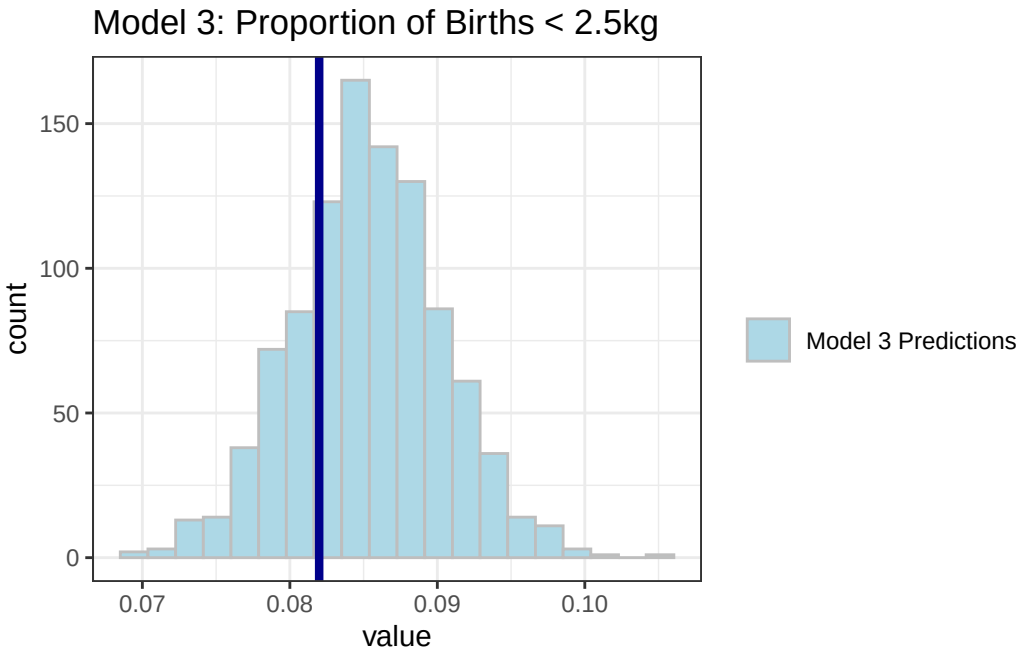
The results indicate that both models perform comparably across education levels, although they seem less accurate for individuals with higher education.

Lastly, we revisit the test statistic of the proportion of births under 2.5kg to compare Model 2 and Model 3:

```
# Test statistics for the original data
obs_proportion_under_2_5kg <- mean(ds$log_weight <= log(2.5))

# Test statistics for posterior predictive simulations
sim_proportion_under_2_5kg_mod2 <- sapply(1:nrow(yrep_mod2), function(i) mean(yrep_mod2[i,
sim_proportion_under_2_5kg_mod3 <- sapply(1:nrow(yrep_mod3), function(i) mean(yrep_mod3[i,

# Histogram plots for the test statistics of both models
ggplot(as_tibble(sim_proportion_under_2_5kg_mod2), aes(x = value)) +
  geom_histogram(aes(fill = "Model 2 Predictions"), bins = 20, color = "grey") +
  geom_vline(xintercept = obs_proportion_under_2_5kg, color = "darkblue", lwd = 1.5) +
  ggtitle("Model 2: Proportion of Births < 2.5kg") +
  theme_bw() +
  scale_fill_manual(name = "", values = c("Model 2 Predictions" = "lightblue"))
```



```
ggplot(as_tibble(sim_proportion_under_2_5kg_mod3), aes(x = value)) +
  geom_histogram(aes(fill = "Model 3 Predictions"), bins = 20, color = "grey") +
  geom_vline(xintercept = obs_proportion_under_2_5kg, color = "darkblue", lwd = 1.5) +
  ggtitle("Model 3: Proportion of Births < 2.5kg") +
  theme_bw() +
  scale_fill_manual(name = "", values = c("Model 3 Predictions" = "lightblue"))
```



Model 3 shows a slightly more concentrated distribution around the observed test statistic, which may suggest a slight improvement over Model 2.

Upon completion of all checks, it appears that the addition of the "sex" variable to Model 2 has offered a very limited imporvement.

```
  fit3 <- extract(mod3)
  loglik3 <- fit3$log_lik
  loo3 <- loo(loglik3, save_psis = TRUE)
  loo_comparison <- loo_compare(loo2, loo3)
  loo_comparison
```

```
       elpd_diff se_diff
model2  0.0       0.0
model1 -0.5       0.3
```

As expected, model 3 is a bit better than model 2 by imporve elpd_diff by 0.5.