Week 6: Visualizing the Bayesian Workflow

27/02/23

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(here("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
```

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:

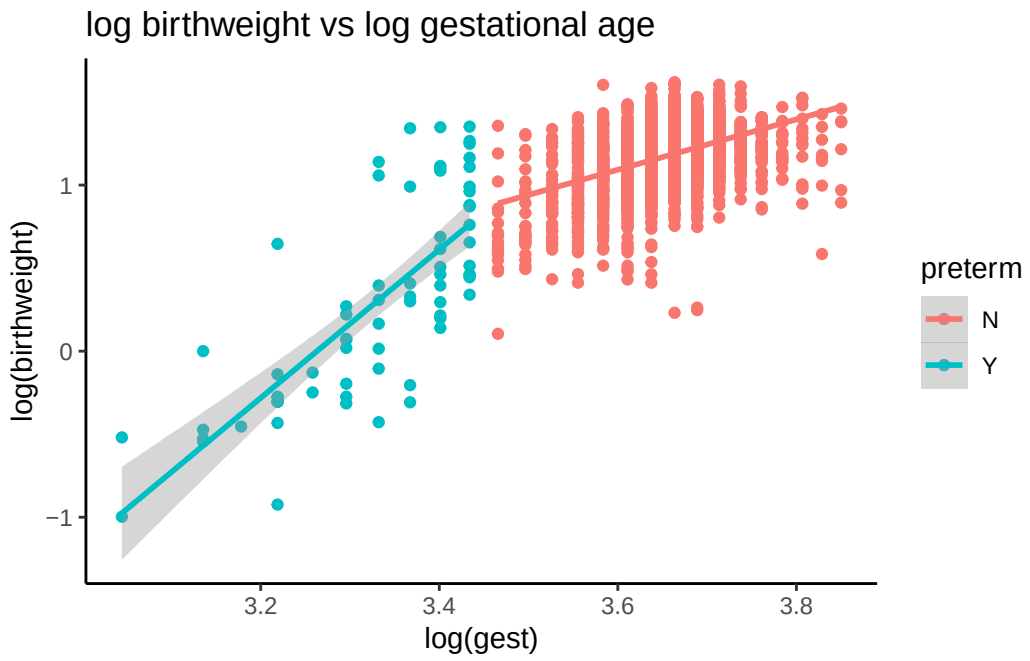
- Explain what your graph/ tables show
- Choose a graph type that's appropriate to the data type
- If you use `geom_smooth`, please also plot the underlying data

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.

I will first replicate the result from our lecture:

```
ds |>
  ggplot(aes(log(gest), log(birthweight), color = preterm)) +
  geom_point() +
```

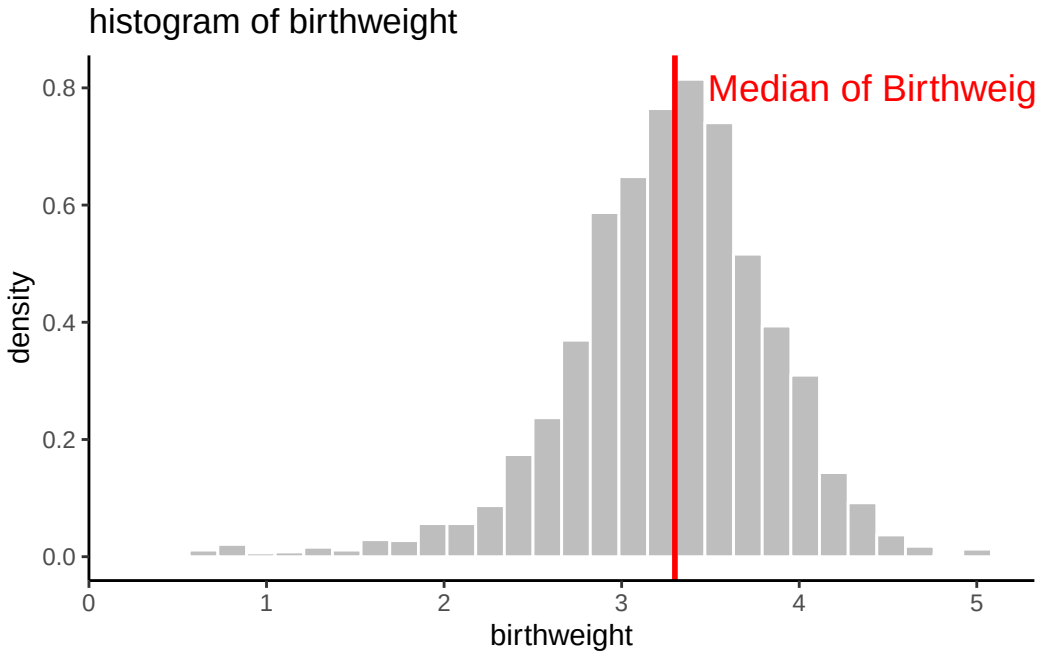
```
  geom_smooth(method = 'lm') +
  theme_classic() +
  labs(title = 'log birthweight vs log gestational age')
```



as we can see, it is quite clear that for both preterm group No and Yes, the birthweight is positively correlated with gestational age.

second let's plot the histogram of the birthweight, to later compare with our simulated results:

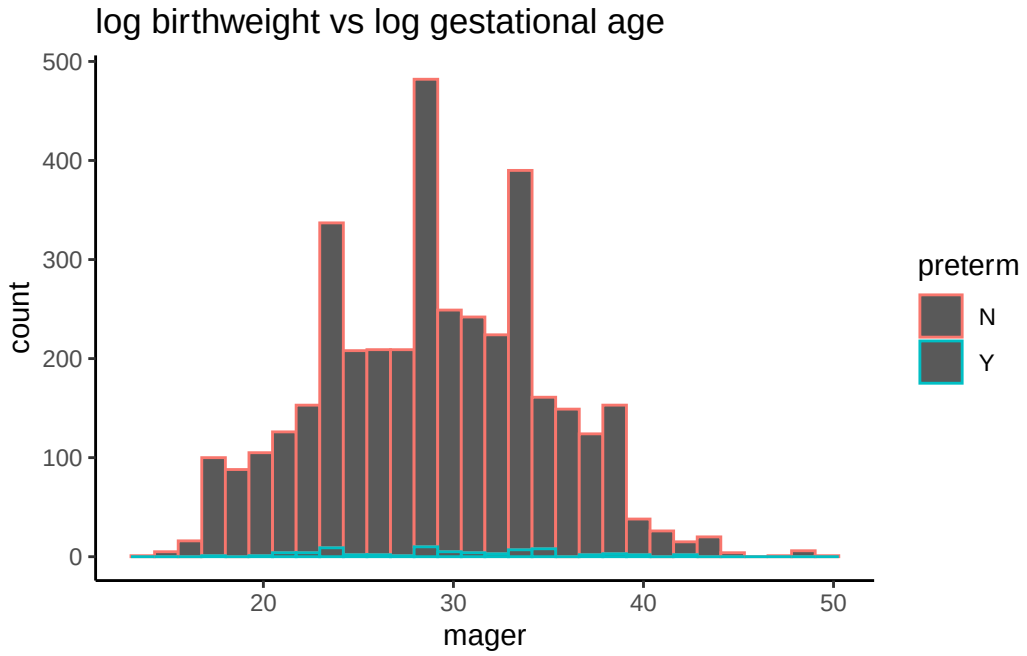
```
ds |>
  ggplot(aes(x = birthweight, y = after_stat(density))) +
  geom_histogram(bins = 30, color = 'white', fill = 'gray') +
  geom_vline(aes(xintercept = median(birthweight)), color = "red", linewidth = 1)+
  theme_classic() +
  annotate("text", x=4.5, y=0.8, label= "Median of Birthweight",
           color = "red", size = 5) +
  labs(title = 'histogram of birthweight')
```



As we can see from the above histogram, the birthweight of children is approximately normal distributed, with a median around 3.25 (which could later be used as the statistics on simulation), and with very few occurrence of weight less than 1.5 or greater than 5.

let's also draw a histogram of mom's age with correspondence to the preterm Yes/No, to see if there's any pattern between them:

```
ds |>
  ggplot(aes(mager, color = preterm)) +
  geom_histogram() +
  theme_classic() +
  labs(title = 'log birthweight vs log gestational age')
```



As we can see, the three most frequent age group of the participant is about 24, 29, and 34, with the Preterm No being the most frequent, and Preterm Yes has no significant correlation with mom's age.

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.

For reference, I was reading Professor's blog at: https://www.monicaalexander.com/posts/2020-28-02-bayes_viz/

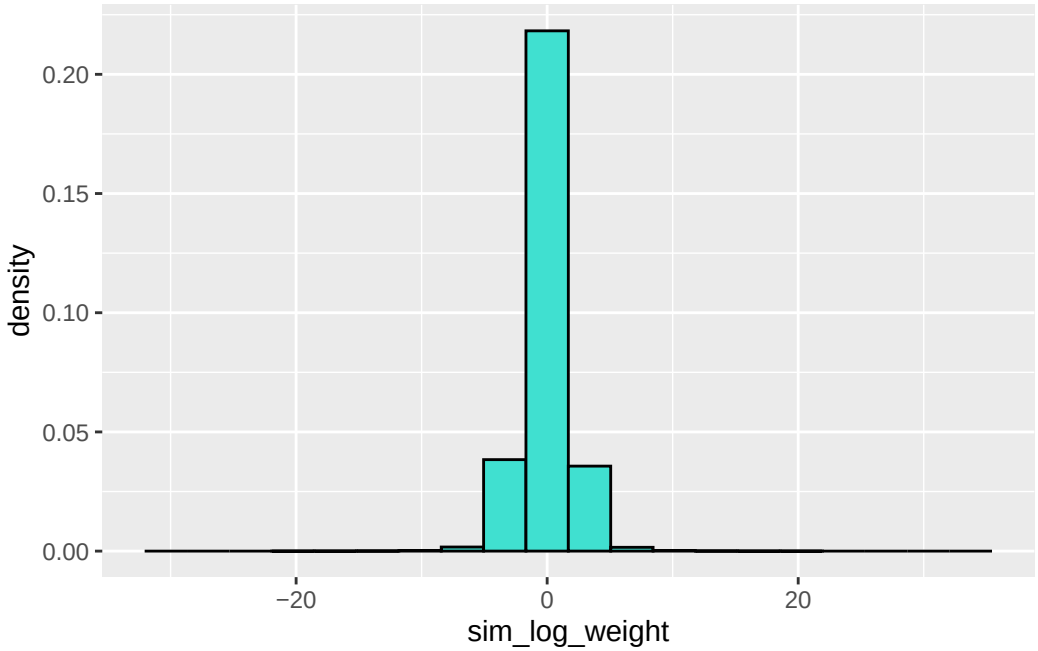
let's set up simulation first:

```
n <- 1000

b0 <- rnorm(n, 0, 1)
b1 <- rnorm(n, 0, 1)

sigma <- abs(rnorm(n, 0, 1))

# ds$log_gest_c = (log(ds$gest)-mean(log(ds$gest)))/sd(log(ds$gest))

sim <- tibble(log_gest_c = (log(ds$gest)-mean(log(ds$gest)))/sd(log(ds$gest)))

d <- nrow(sim)
```

```
for (i in 1:n){
  mu_i <- b0[i] + b1[i] * sim$log_gest_c
  sim[paste0(i)] <- mu_i + rnorm(d, 0 , sigma[i])
}
```

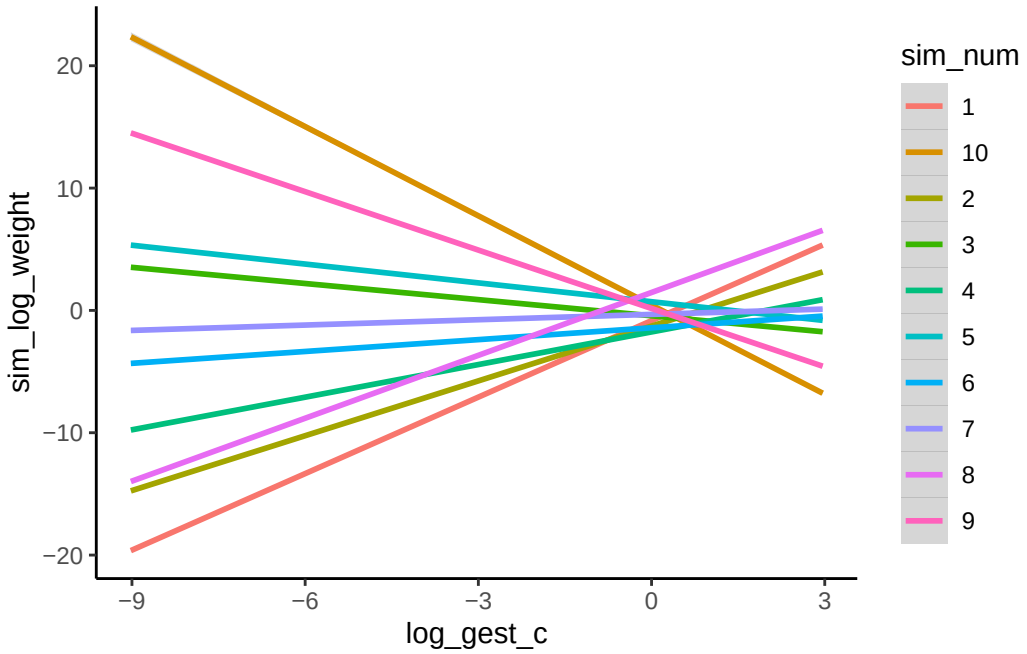
- Plot the resulting distribution of simulated (log) birth weights.

```
sim |>
  pivot_longer(`1`:`1000`, names_to = 'sim_num', values_to = 'sim_log_weight') |>
  ggplot(aes(sim_log_weight)) +
  geom_histogram(aes(y = after_stat(density)), bins = 20, fill = "turquoise", color = "bla
```



- Plot ten simulations of (log) birthweights against gestational age.

```
sim |>
  select(log_gest_c,`1`:`10`) |>
  pivot_longer(`1`:`10`, names_to = 'sim_num', values_to = 'sim_log_weight') |>
  ggplot(aes(log_gest_c, sim_log_weight, color = sim_num)) +
  geom_smooth(method = 'lm') +
  theme_classic()
```



Not so sure if this is something we are looking for…but as we can see, even with informative priors, the linear plot still can diverge to many different directions.

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

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

```
            seed = 243)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
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/4.2-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/Rc
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
              ^
              ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'simple_weight' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.00013 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.3 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: 0.319543 seconds (Warm-up)
Chain 1:                0.275598 seconds (Sampling)
Chain 1:                0.595141 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'simple_weight' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000112 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.12 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: 0.312617 seconds (Warm-up)
Chain 2:                0.287994 seconds (Sampling)
Chain 2:                0.600611 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'simple_weight' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000111 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.11 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: 0.310057 seconds (Warm-up)
Chain 3:                0.255554 seconds (Sampling)
Chain 3:                0.565611 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'simple_weight' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000111 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.11 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: 0.305027 seconds (Warm-up)
Chain 4:                0.267133 seconds (Sampling)
Chain 4:                0.57216 seconds (Total)
Chain 4:
```

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

```
             mean      se_mean          sd      2.5%       25%       50%
beta[1] 1.1625515 7.177826e-05 0.002588909 1.1574785 1.1609120 1.1625568
beta[2] 0.1437529 7.621667e-05 0.002649394 0.1384928 0.1419098 0.1438130
sigma   0.1688257 1.128346e-04 0.002080365 0.1647921 0.1674232 0.1687016
              75%     97.5%     n_eff      Rhat
beta[1] 1.1641243 1.1678590 1300.9108 0.9982284
beta[2] 0.1455522 0.1491357 1208.3515 1.0011315
sigma   0.1702699 0.1729892  339.9336 1.0088640
```

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.

The model is:

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

plug them in we have

```r
log_gest_37 <- (log(37) - mean(log(ds$gest)))/sd(log(ds$gest))
log_expect <- 1.1625515 + 0.1437529 * log_gest_37
expect <- exp(log_expect)
expect
```

```
[1] 2.936089
```

the expected birthweight of a baby who was born at a gestational age of 37 weeks is 2.94 kg

Question 4

Write a stan model to run Model 2, and run it.

```r
# following piazza questions, converting preterm to 0, 1
preterm <- ifelse(ds$preterm=="Y", 1, 0)

stan_data <- list(N = nrow(ds),
                  log_weight = ds$log_weight,
                  log_gest = ds$log_gest_c,
                  preterm = preterm)
```

```r
mod2 <- stan(data = stan_data,
             file = "/Users/haochensong/Desktop/Winter 2023/Applied Statistics II/Applied-
             iter = 500,
             seed = 243)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
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/4.2-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/Rc
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'model2' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000405 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 4.05 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: 1.27298 seconds (Warm-up)
Chain 1:                0.998638 seconds (Sampling)
Chain 1:                2.27162 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'model2' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000258 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.58 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: 1.24405 seconds (Warm-up)
Chain 2:                1.06532 seconds (Sampling)
Chain 2:                2.30937 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'model2' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000256 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.56 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: 1.34428 seconds (Warm-up)
Chain 3:                1.0355 seconds (Sampling)
Chain 3:                2.37977 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'model2' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.00025 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.5 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: 1.25132 seconds (Warm-up)
Chain 4:                1.14013 seconds (Sampling)
Chain 4:                2.39145 seconds (Total)
Chain 4:
```

check my results:

```
summary(mod2)$summary[c("beta[1]", "beta[2]", "beta[3]" , "beta[4]", "sigma"),]
```

```
            mean      se_mean          sd       2.5%        25%       50%
beta[1] 1.1694946 7.572976e-05 0.002711485 1.16409488 1.16767558 1.1695118
beta[2] 0.1020224 1.100593e-04 0.003495701 0.09492474 0.09969273 0.1020986
beta[3] 0.5552154 3.173582e-03 0.060939105 0.42980040 0.51444226 0.5559141
beta[4] 0.1968609 6.451083e-04 0.012680929 0.17253226 0.18795660 0.1963624
sigma   0.1612444 7.560698e-05 0.001815268 0.15776726 0.16004864 0.1612062
             75%     97.5%     n_eff      Rhat
beta[1] 1.1711198 1.1749489 1281.9794 0.9978688
beta[2] 0.1043769 0.1085849 1008.8241 1.0023916
beta[3] 0.5986341 0.6727090  368.7167 1.0187539
beta[4] 0.2057929 0.2206898  386.3998 1.0239494
sigma   0.1624964 0.1648063  576.4449 1.0020277
```

Question 5

For reference I have uploaded some model 2 results. Check your results are similar.

```
#load(here("output", "mod2.Rda"))
#summary(mod2)$summary[c(paste0("beta[", 1:4, "]"), "sigma"),]
```

The results are similar but beta[2] and beta[3] flipped…

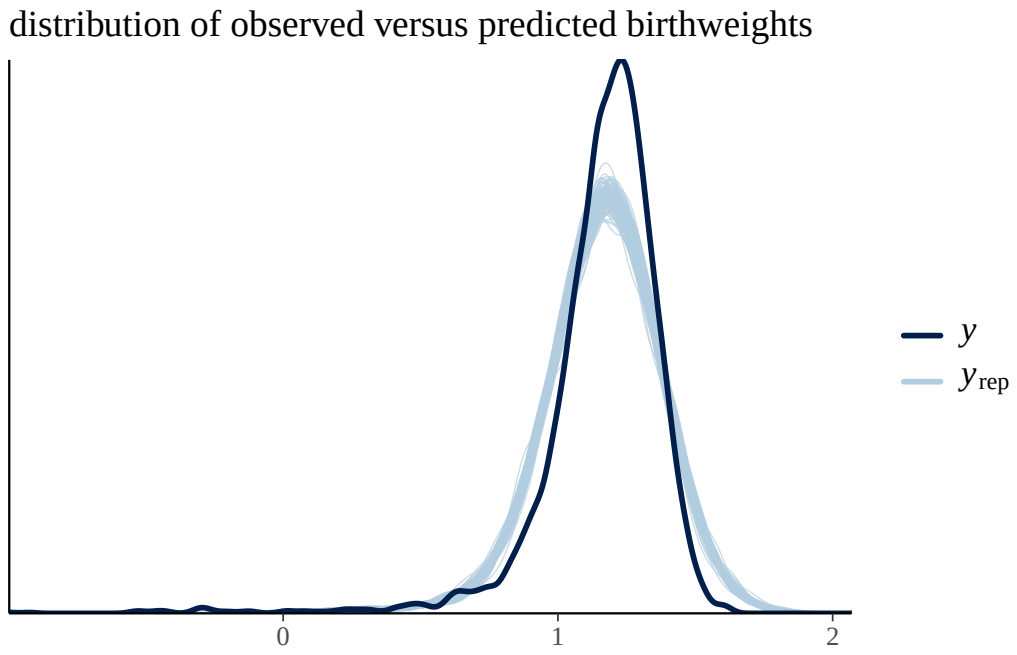
PPCs

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"]]
yrep2 <- extract(mod2)[["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

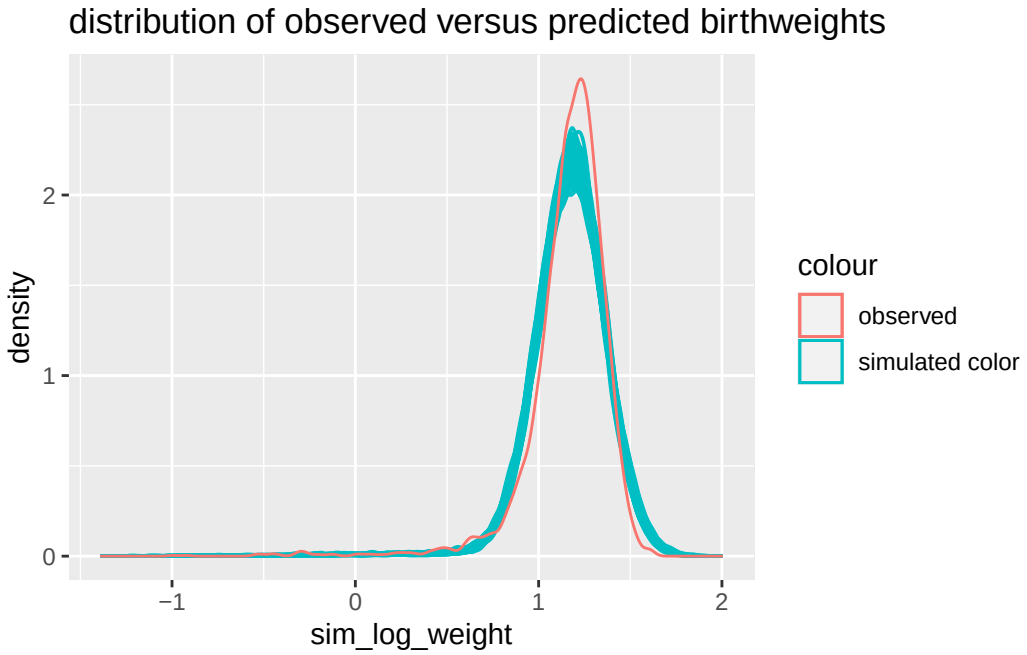
Make a similar plot to the one above but for model 2, and **not** using the bayes plot in built function (i.e. do it yourself just with `geom_density`)

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

```
[1] 1000 3842
```

```
# first let's get them into a tibble

yrep2 = t(yrep2)
```

```r
colnames(yrep2) <- 1:1000


dp <- as_tibble(yrep2) |>
  pivot_longer(`1`:`1000`, names_to = 'sim_num',
               values_to = 'sim_log_weight') |>
  filter(sim_num %in% samp100)

dp |>
  ggplot(aes(sim_log_weight, group = sim_num)) +
  geom_density(alpha = 0.2, aes(color = 'simulated color'))+
  geom_density(data = ds |> mutate(sim_num = 1),
               aes(x = log(birthweight), color = 'observed'))+
  ggtitle("distribution of observed versus predicted birthweights")
```

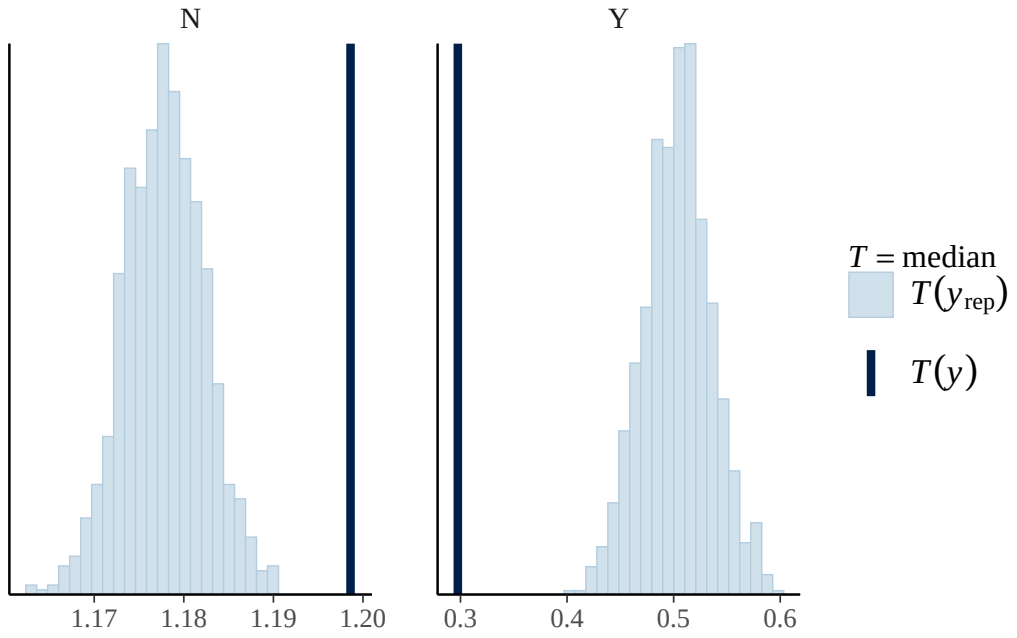


Test statistics

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).

```
y <- ds$log_weight
yrep1 <- extract(mod1)[["log_weight_rep"]]
yrep2 <- extract(mod2)[["log_weight_rep"]]

stat_mod1 <- c()
stat_mod2 <- c()
for (i in 1:1000){
  stat_y <- mean(y<=log(2.5))
  stat_mod1[i] <- mean(yrep1[i,] <= log(2.5))
  stat_mod2[i] <- mean(yrep2[i,] <= log(2.5))
}
```

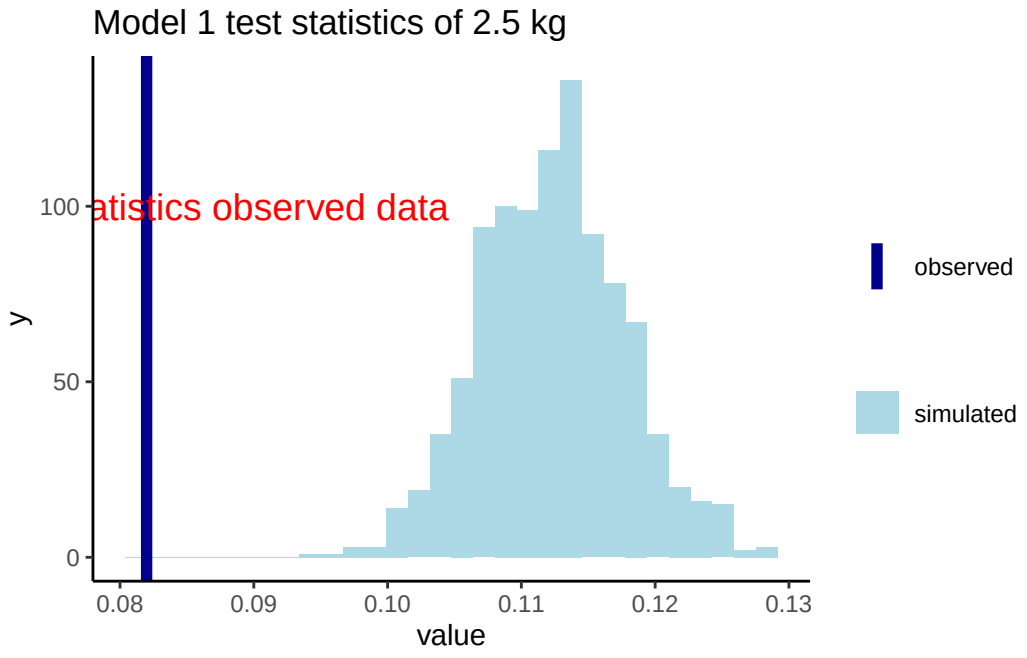
the stat for the data is 0.0820

and the stats for model 1 is stored at stat_mod1

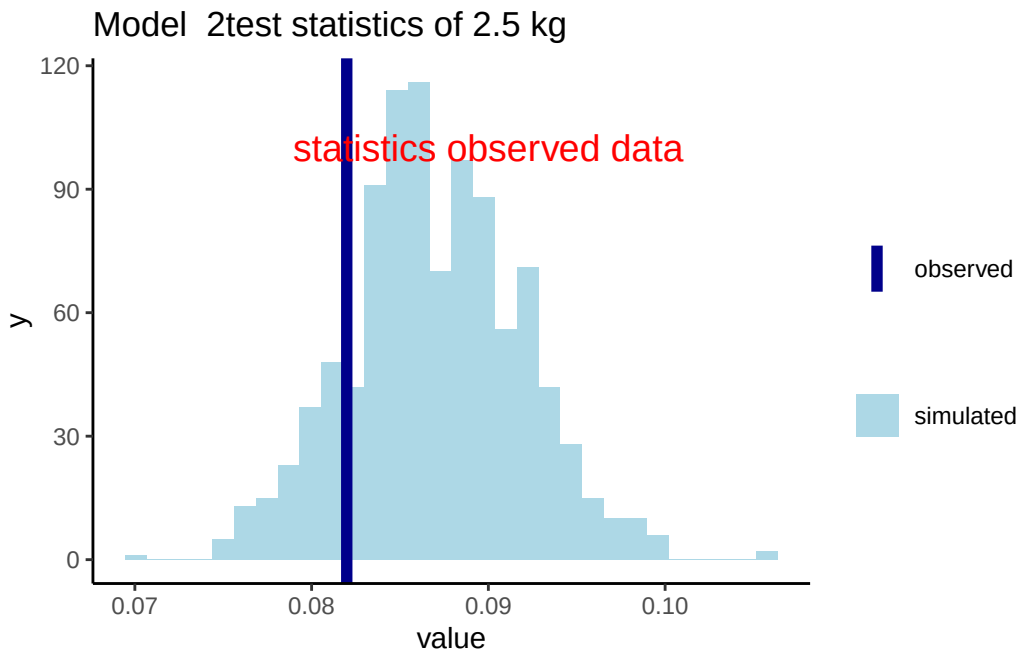
and the stats for model 2 is stored at stat_mod2

let's make the plot for the two models:

```
#Model 1:

stat_mod1 |>
  as.tibble() |>
  ggplot(aes(value)) +
  geom_histogram(aes(fill = 'simulated'))+
  geom_vline(aes(xintercept = stat_y, color = "observed"), lwd = 2)+
  ggtitle('Model 1 test statistics of 2.5 kg')+
  annotate("text", x=0.09, y=100, label= "statistics observed data",
           color = "red", size = 5) +
  scale_color_manual(name = "",
                     values = c("observed" = "darkblue"))+
  scale_fill_manual(name = "",
                     values = c("simulated" = "lightblue")) +
  # the manual change of color idea is adapted from professor's blog post
  theme_classic()
```



```r
#Model 2:

stat_mod2 |>
  as.tibble() |>
  ggplot(aes(value)) +
  geom_histogram(aes(fill = 'simulated'))+
  geom_vline(aes(xintercept = stat_y, color = "observed"), lwd = 2)+
  ggtitle('Model  2test statistics of 2.5 kg')+
  annotate("text", x=0.09, y=100, label= "statistics observed data",
           color = "red", size = 5) +
  scale_color_manual(name = "",
                     values = c("observed" = "darkblue"))+
  scale_fill_manual(name = "",
                     values = c("simulated" = "lightblue")) +

  # the manual change of color idea is adapted from professor's blog post
  theme_classic()
```



Therefore as we can see from the two plots, model 2 seems to perform better.

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"]]
loglik2 <- extract(mod2)[["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)
loo2 <- loo(loglik2, 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   1.3
looic     -2754.9 145.2
------
Monte Carlo SE of elpd_loo is 0.1.

All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

```
loo2
```

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

         Estimate    SE
elpd_loo   1553.2  69.8
```

```
p_loo        14.3   2.1
looic     -3106.5 139.5
------
Monte Carlo SE of elpd_loo is 0.1.

All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

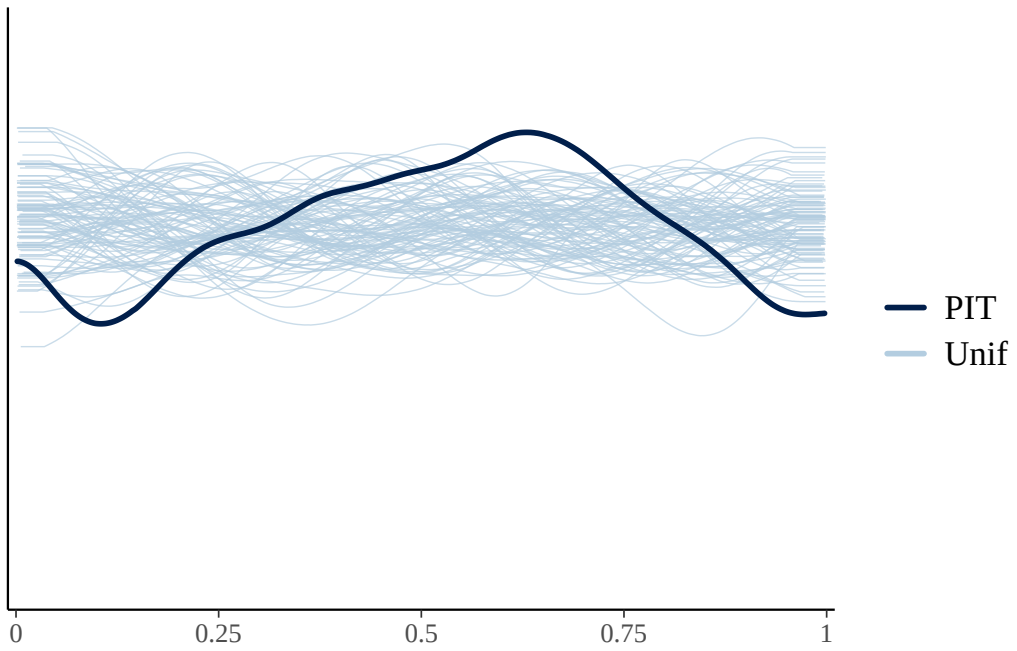
Comparing the two models tells us Model 2 is better:

```
loo_compare(loo1, loo2)
```

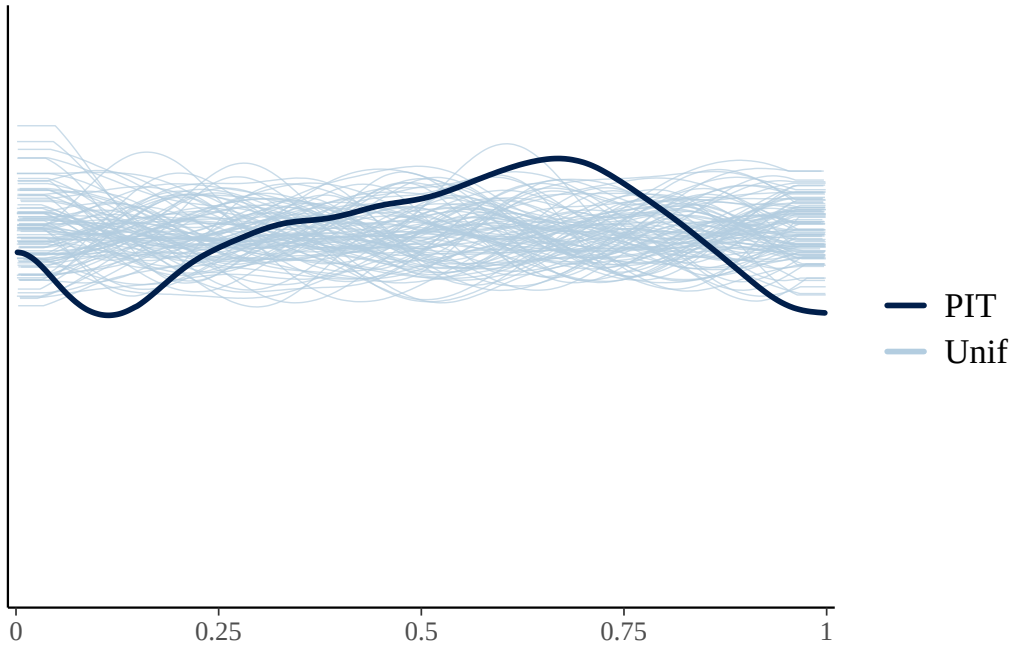
```
       elpd_diff se_diff
model2    0.0       0.0
model1 -175.8      36.3
```

We can also compare the LOO-PIT of each of the models to standard uniforms. The both do pretty well.

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



```r
ppc_loo_pit_overlay(yrep = yrep2, y = y, lw = weights(loo2$psis_object))
```



Bonus question (not required)

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

Question 8

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.

let's add the effect of age into the model, that is:

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

Let's call this model 3:

```r
# following piazza questions, converting preterm to 0, 1
preterm <- ifelse(ds$preterm=="Y", 1, 0)

stan_data <- list(N = nrow(ds),
```

```
                  log_weight = ds$log_weight,
                  log_gest = ds$log_gest_c,
                  preterm = preterm,
                  log_age = log(ds$mager))
mod3 <- stan(data = stan_data,
             file = "/Users/haochensong/Desktop/Winter 2023/Applied Statistics II/Applied-
             iter = 500,
             seed = 243)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
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/4.2-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/Rc
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
              ^
              ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'model3' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000468 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 4.68 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)
```

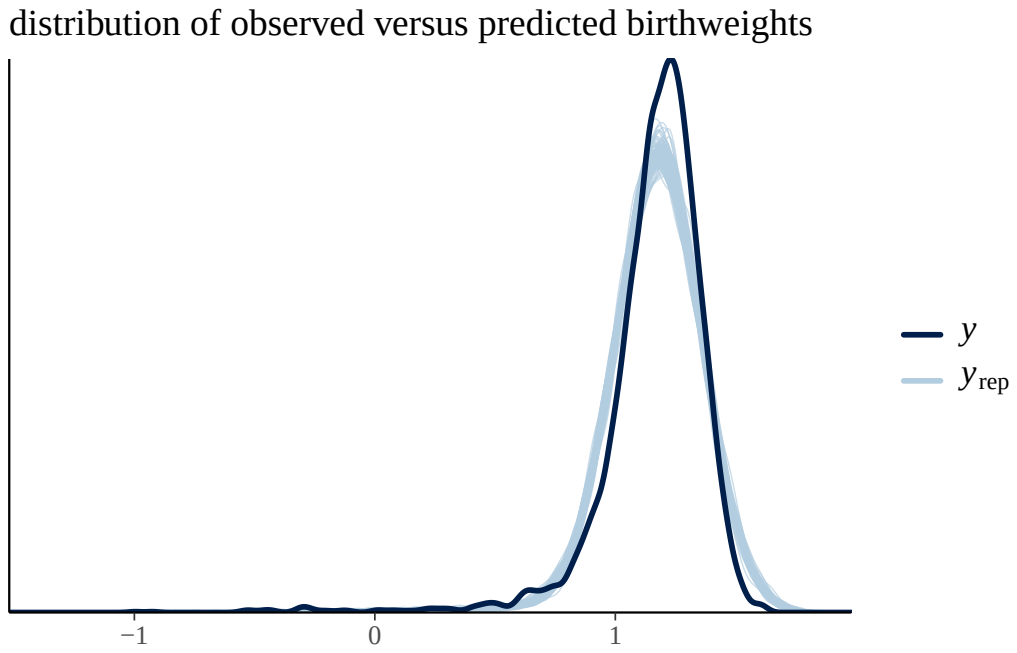
```
Chain 1: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 1: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 1: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 1: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 1: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 1: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 1: 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: 5.21634 seconds (Warm-up)
Chain 1:                5.14453 seconds (Sampling)
Chain 1:                10.3609 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'model3' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000334 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 3.34 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: 8.40386 seconds (Warm-up)
Chain 2:                6.44301 seconds (Sampling)
Chain 2:                14.8469 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'model3' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000324 seconds
```

```
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 3.24 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: 7.33783 seconds (Warm-up)
Chain 3:                5.73536 seconds (Sampling)
Chain 3:                13.0732 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'model3' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000323 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 3.23 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: 5.66527 seconds (Warm-up)
```

```
Chain 4:                6.19174 seconds (Sampling)
Chain 4:                11.857 seconds (Total)
Chain 4:
```

let's see how model 3 replicates the results:

```
set.seed(1856)
y <- ds$log_weight
yrep1 <- extract(mod1)[["log_weight_rep"]]
yrep2 <- extract(mod2)[["log_weight_rep"]]
yrep3 <- extract(mod3)[["log_weight_rep"]]
samp100 <- sample(nrow(yrep3), 100)
ppc_dens_overlay(y, yrep3[samp100, ])  +
  ggtitle("distribution of observed versus predicted birthweights")
```



I think it is comparable to model 2.

Let's look at the summary of model3:

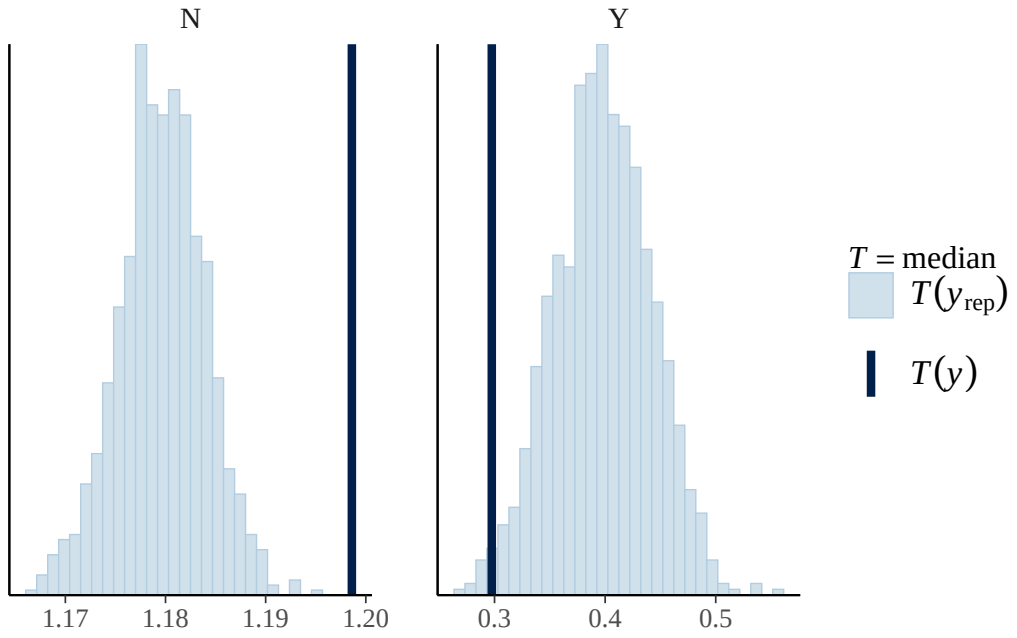
```
summary(mod3)$summary[c("beta[1]", "beta[2]", "beta[3]" , "beta[4]", "beta[5]", "sigma"),]
```

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

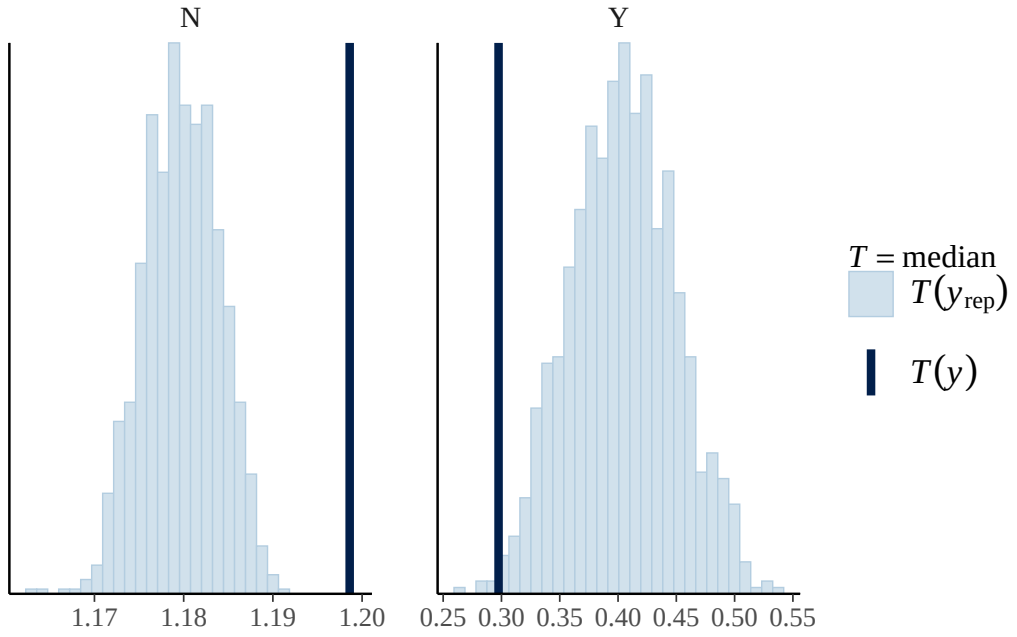
```
beta[1] 0.92436314 2.045785e-03 0.041527417 0.84690197 0.89453574 0.92338486
beta[2] 0.10258379 1.078803e-04 0.003580324 0.09588042 0.10005248 0.10270519
beta[3] 0.56861865 3.091109e-03 0.063821347 0.43472617 0.52787382 0.56874844
beta[4] 0.19942078 6.670581e-04 0.013259432 0.17257413 0.19123213 0.19950029
beta[5] 0.07332901 6.121591e-04 0.012333537 0.04858707 0.06499382 0.07352043
sigma   0.16046051 6.493401e-05 0.001805521 0.15697536 0.15926372 0.16042246
              75%      97.5%     n_eff      Rhat
beta[1] 0.9523140 1.00821385  412.0498 1.0048515
beta[2] 0.1050799 0.10929177 1101.4386 0.9984528
beta[3] 0.6106322 0.69601099  426.2881 1.0046883
beta[4] 0.2080637 0.22444184  395.1141 1.0035208
beta[5] 0.0820781 0.09662455  405.9259 1.0048669
sigma   0.1616716 0.16393523  773.1445 0.9987167
```

let's compare on two statistics: first on median for prematurity:

```
ppc_stat_grouped(ds$log_weight, yrep2, group = ds$preterm, stat = 'median')
```




```
ppc_stat_grouped(ds$log_weight, yrep3, group = ds$preterm, stat = 'median')
```

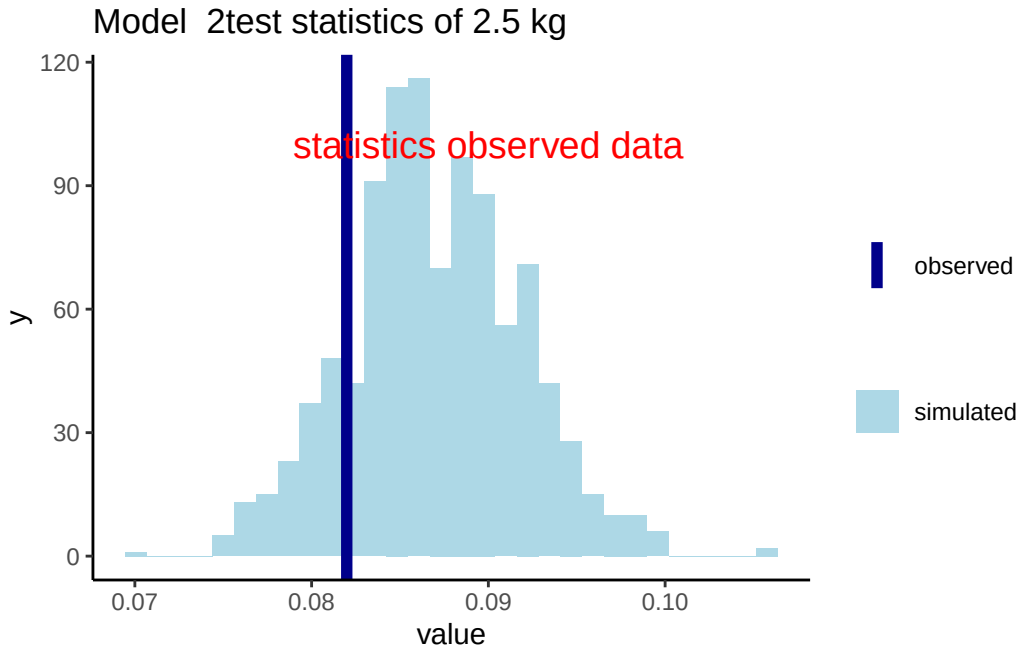


Secondly on birthweight less than 2.5 kg:

```
y <- ds$log_weight
yrep1 <- extract(mod1)[["log_weight_rep"]]
yrep2 <- extract(mod2)[["log_weight_rep"]]

stat_mod1 <- c()
stat_mod2 <- c()
stat_mod3 <- c()
for (i in 1:1000){
  stat_y <- mean(y<=log(2.5))
  stat_mod1[i] <- mean(yrep1[i,] <= log(2.5))
  stat_mod2[i] <- mean(yrep2[i,] <= log(2.5))
  stat_mod3[i] <- mean(yrep3[i,] <= log(2.5))
}
```

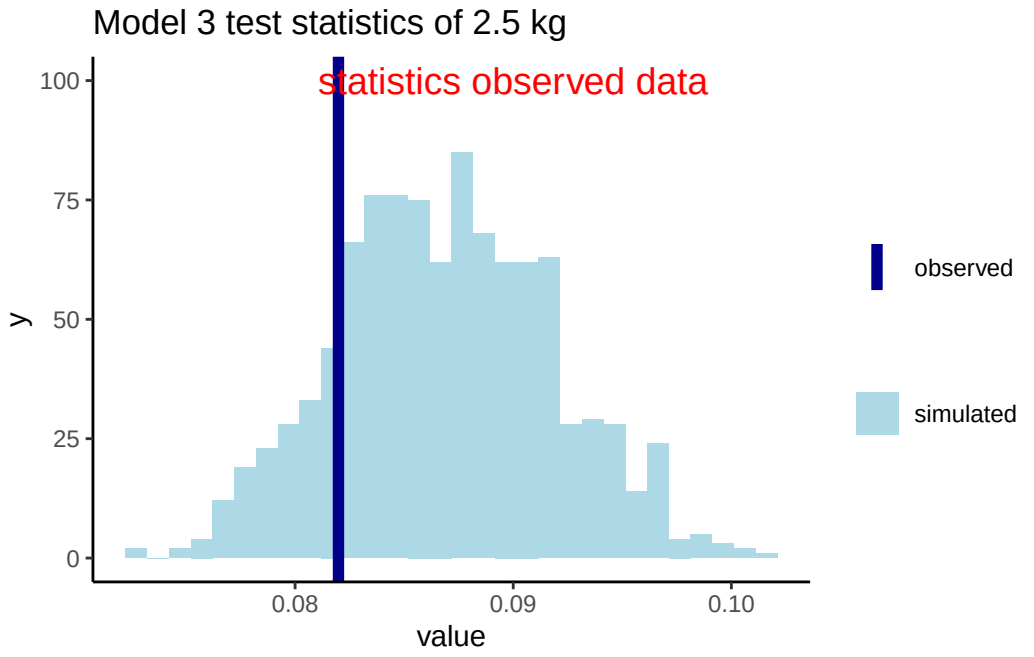
```
#Model 2:

stat_mod2 |>
  as.tibble() |>
  ggplot(aes(value)) +
  geom_histogram(aes(fill = 'simulated'))+
  geom_vline(aes(xintercept = stat_y, color = "observed"), lwd = 2)+
```

```
  ggtitle('Model  2test statistics of 2.5 kg')+
  annotate("text", x=0.09, y=100, label= "statistics observed data",
           color = "red", size = 5) +
  scale_color_manual(name = "",
                     values = c("observed" = "darkblue"))+
  scale_fill_manual(name = "",
                     values = c("simulated" = "lightblue")) +
  # the manual change of color idea is adapted from professor's blog post
  theme_classic()
```




```
#Model 3:

stat_mod3 |>
  as.tibble() |>
  ggplot(aes(value)) +
  geom_histogram(aes(fill = 'simulated'))+
  geom_vline(aes(xintercept = stat_y, color = "observed"), lwd = 2)+
  ggtitle('Model 3 test statistics of 2.5 kg')+
  annotate("text", x=0.09, y=100, label= "statistics observed data",
           color = "red", size = 5) +
  scale_color_manual(name = "",
                     values = c("observed" = "darkblue"))+
```

```
scale_fill_manual(name = "",
                  values = c("simulated" = "lightblue")) +
# the manual change of color idea is adapted from professor's blog post
theme_classic()
```



as we can see, with adding the covariate of log(age), the statistics of proportion less than 2.5kg in model 3 performs slightly better than model 2