STA2201_Lab6

Alice Huang

15/02/2023

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("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)
```

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

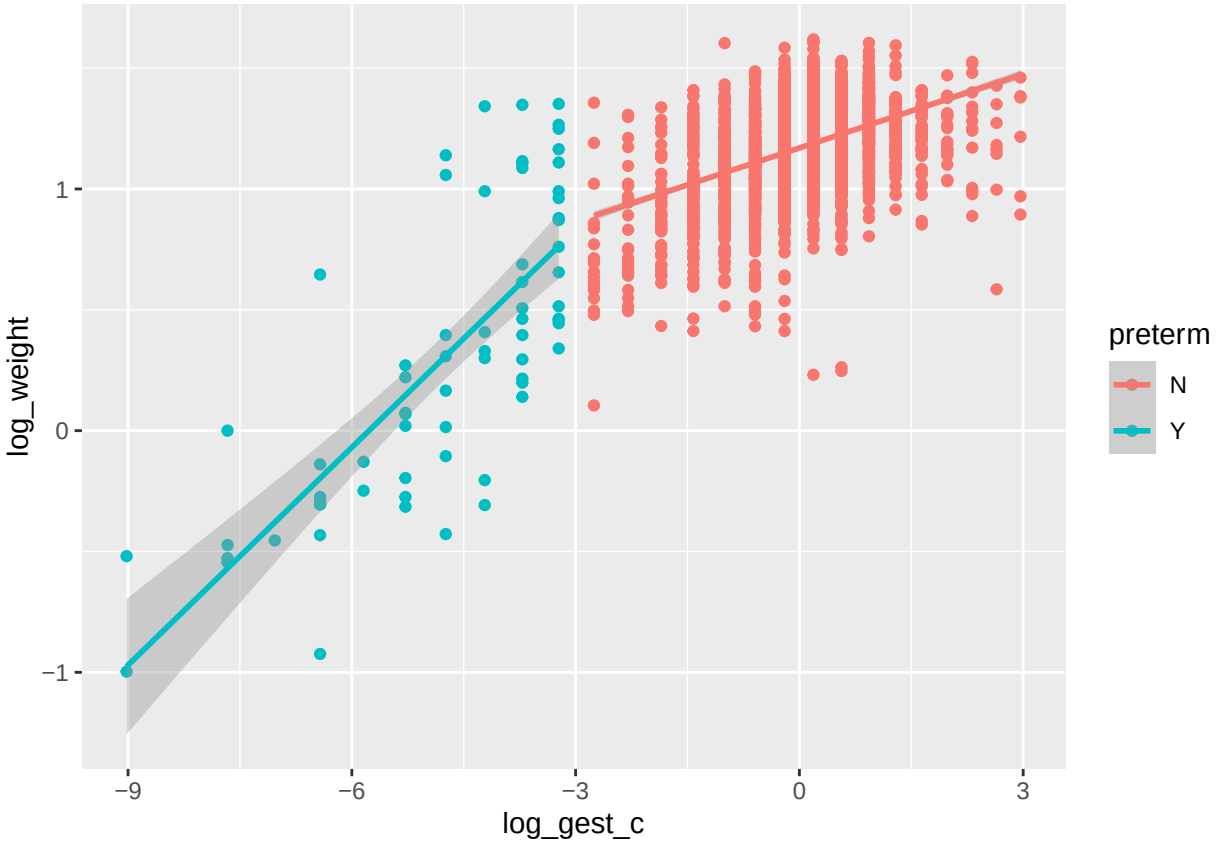
Question 1

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

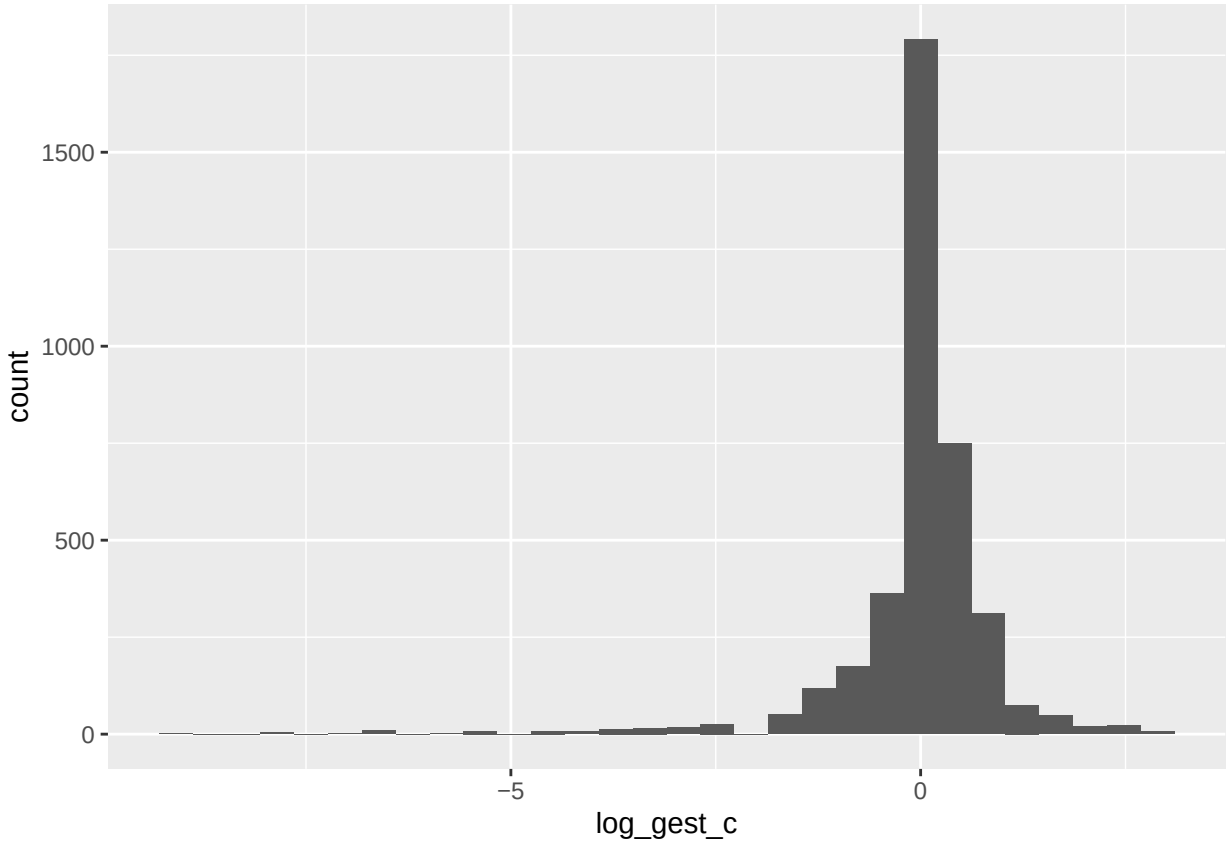
Feel free to replicate one of the scatter plots in the lectures as one of the interesting observations, as those form the basis of our models.

```
ds %>% ggplot(aes(x=log_gest_c, y=log_weight, color=preterm)) + geom_point() + geom_smooth(method="lm")
```



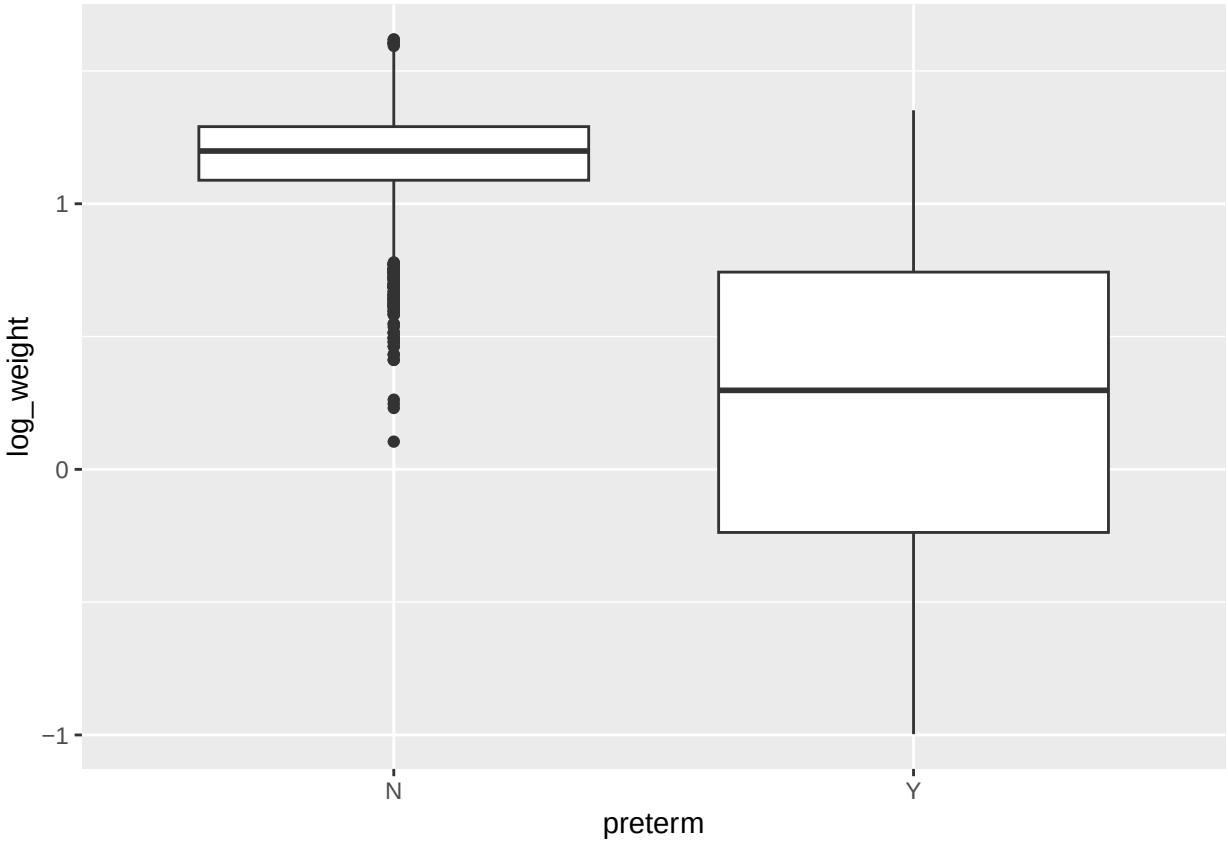
We see that as the log of gestational age increases the log of the birthweight increases. However, this increase seems to be greater for babies who were born prematurely rather than babies who were not born prematurely.

```
ds %>% ggplot(aes(x=log_gest_c)) + geom_histogram()
```



We can see that the log of the gestational age follows approximately a normal distribution with mean 0

```
ds %>% ggplot(aes(x=preterm, y=log_weight)) + geom_boxplot()
```



We can see that the median log of birthweight is higher for non-premature babies than for premature babies. However, the standard deviation of log birthweight for premature babies is much higher than it is for non premature babies.

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.

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.

```
# 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("simple_weight.stan"),
             iter = 500,
             seed = 243)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000325 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 3.25 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: 0.633 seconds (Warm-up)
## Chain 1:                0.502 seconds (Sampling)
## Chain 1:                1.135 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000247 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.47 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: 0.538 seconds (Warm-up)
## Chain 2:                0.513 seconds (Sampling)
## Chain 2:                1.051 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000205 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.05 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: 0.546 seconds (Warm-up)
## Chain 3:                0.514 seconds (Sampling)
## Chain 3:                1.06 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000404 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 4.04 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: 0.564 seconds (Warm-up)
## Chain 4:                0.443 seconds (Sampling)
## Chain 4:                1.007 seconds (Total)
## Chain 4:
```

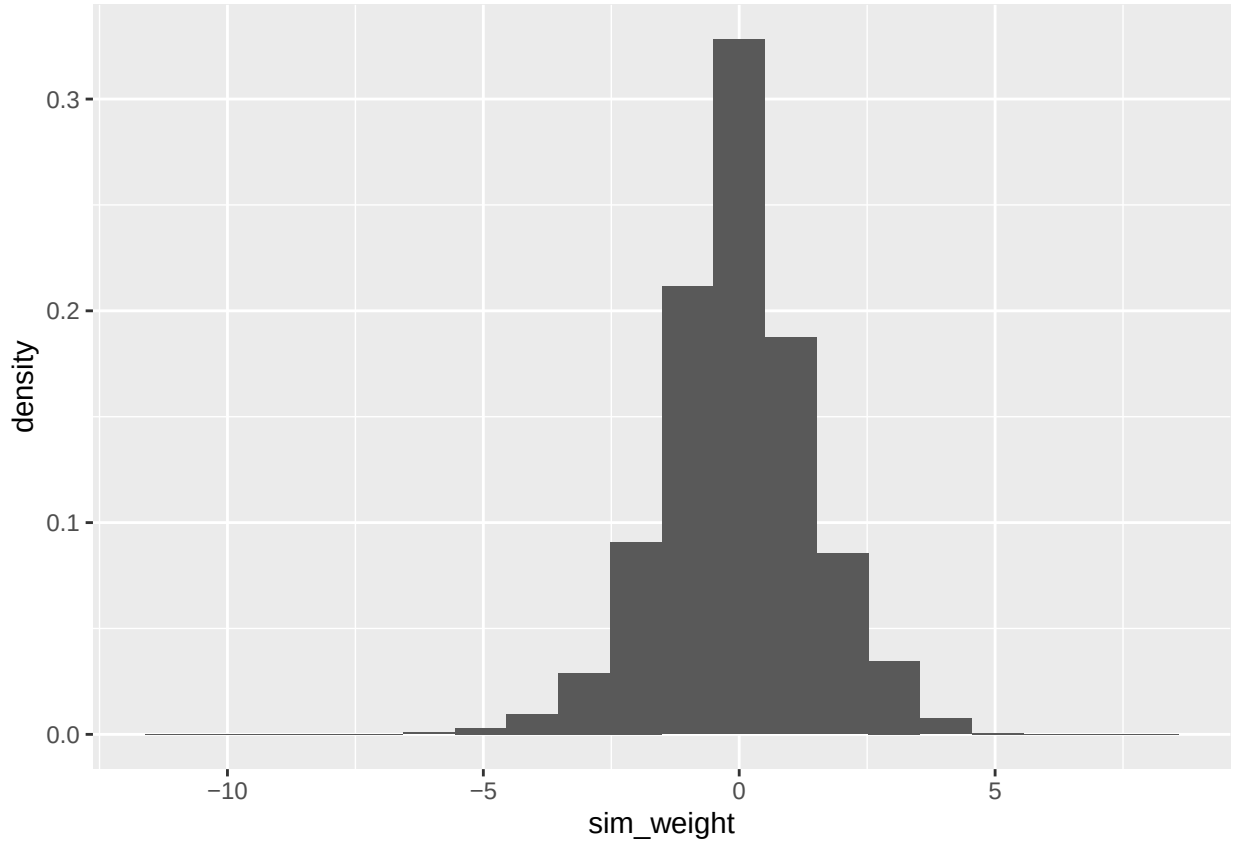
```
summary(mod1)$summary[c("beta[1]", "beta[2]", "sigma"),]
```

```
##              mean      se_mean          sd      2.5%       25%       50%
## beta[1] 1.1624783 8.160385e-05 0.002856578 1.1570200 1.1604786 1.1625011
## beta[2] 0.1437529 8.295075e-05 0.002912236 0.1381284 0.1416970 0.1436747
## sigma   0.1690330 1.113724e-04 0.001902828 0.1652694 0.1677842 0.1690763
##              75%     97.5%     n_eff      Rhat
## beta[1] 1.1644669 1.1681028 1225.3801 0.9978044
## beta[2] 0.1456716 0.1495180 1232.5721 0.9998714
## sigma   0.1702528 0.1727953  291.9066 1.0146111
```

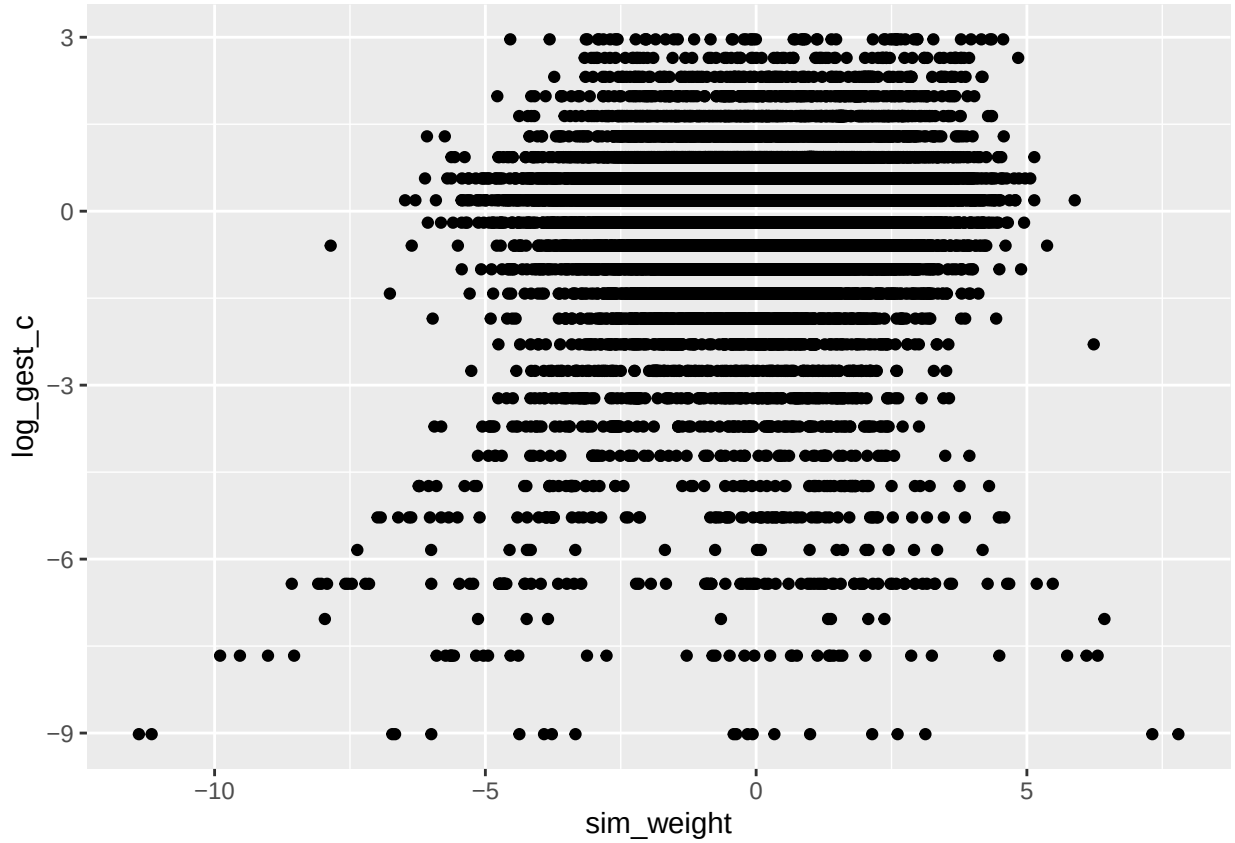
1% change in standardized birthweight leads to 0.14% change in .. (x is log and y is log)

```
nsims <- 1000

sigma <- abs(rnorm(nsims, 0, 1))
beta0 <- rnorm(nsims, 0, 1)
beta1 <- rnorm(nsims, 0, 1)
```

```r
dsims <- tibble(log_gest_c = (log(ds$gest)-mean(log(ds$gest)))/sd(log(ds$gest)))

for(i in 1:nsims){
  this_mu <- beta0[i] + beta1[i]*dsims$log_gest_c
  dsims[paste0(i)] <- this_mu + rnorm(nrow(dsims), 0, sigma[i])
}

#dsims each row corresponds to an observation and has 1000 simulations

dsl <- dsims %>%
  pivot_longer(`1`:`10`, names_to = "sim", values_to = "sim_weight")

dsl %>%
  ggplot(aes(sim_weight)) + geom_histogram(aes(y = after_stat(density)), bins = 20)
```




```r
dsl10 <- dsims[,1:11] %>%
  pivot_longer(`1`:`10`, names_to = "sim", values_to = "sim_weight")

dsl10 %>%
  ggplot(aes(x=sim_weight, y=log_gest_c)) + geom_point()
```



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.

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

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

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

```
##              mean      se_mean          sd      2.5%      25%       50%
## beta[1] 1.1624783 8.160385e-05 0.002856578 1.1570200 1.160479 1.1625011
## beta[2] 0.1437529 8.295075e-05 0.002912236 0.1381284 0.141697 0.1436747
##               75%    97.5%    n_eff      Rhat
## beta[1] 1.1644669 1.168103 1225.380 0.9978044
## beta[2] 0.1456716 0.149518 1232.572 0.9998714
```

So $E[\log(y_i)] = \beta_1 + \beta_2 \log(x_i)$

The expected log birthweight is $1.1624783 + 0.1437529 \log(37) \approx 1.681558$

So the expected birthweight is $\exp(1.681558) = 5.374$.

Question 4

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

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

```
ds %>% mutate(preterm_f = ifelse(preterm == "N", 0, 1)) %>%
  mutate(log_gest_preterm = log_gest_c*preterm_f) -> ds

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

Now we fit model 2.

```
mod2 <- stan(data = stan_data2,
             file = here("simple_weight2.stan"),
             iter = 500,
             seed = 243)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.003857 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 38.57 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: 3.578 seconds (Warm-up)
## Chain 1:                2.963 seconds (Sampling)
## Chain 1:                6.541 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000886 seconds
```

```
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 8.86 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: 4.162 seconds (Warm-up)
## Chain 2:                3.399 seconds (Sampling)
## Chain 2:                7.561 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000629 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 6.29 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: 4.055 seconds (Warm-up)
## Chain 3:                3.666 seconds (Sampling)
## Chain 3:                7.721 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.001203 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 12.03 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: 5.968 seconds (Warm-up)
## Chain 4:                2.539 seconds (Sampling)
## Chain 4:                8.507 seconds (Total)
## Chain 4:
```

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

```
##              mean      se_mean          sd       2.5%        25%       50%
## beta[1] 1.1697151 7.569377e-05 0.002649784 1.16446450 1.16805660 1.1696477
## beta[2] 0.1019637 1.154683e-04 0.003510611 0.09542475 0.09953286 0.1020110
## beta[3] 0.5583261 2.911643e-03 0.060320817 0.43958173 0.51947380 0.5562372
## beta[4] 0.1974737 6.133293e-04 0.012574769 0.17211526 0.18927590 0.1968525
## sigma   0.1613332 8.179003e-05 0.001912132 0.15761582 0.16004178 0.1612498
##               75%     97.5%     n_eff      Rhat
## beta[1] 1.1715116 1.1749015 1225.4647 1.0002691
## beta[2] 0.1041422 0.1086862  924.3571 0.9996143
## beta[3] 0.5976064 0.6769458  429.1983 1.0036710
## beta[4] 0.2056546 0.2219116  420.3514 1.0049057
## sigma   0.1625844 0.1650502  546.5565 1.0082072
```

Question 5

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

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

```
##              mean      se_mean          sd       2.5%        25%       50%
## beta[1] 1.1697241 1.385590e-04 0.002742186 1.16453578 1.16767109 1.1699278
## beta[2] 0.5563133 5.835253e-03 0.058054991 0.43745504 0.51708255 0.5561553
## beta[3] 0.1020960 1.481816e-04 0.003669476 0.09459462 0.09997153 0.1020339
## beta[4] 0.1967671 1.129799e-03 0.012458398 0.17164533 0.18817091 0.1974114
## sigma   0.1610727 9.950037e-05 0.001782004 0.15784213 0.15978020 0.1610734
##               75%     97.5%     n_eff      Rhat
## beta[1] 1.1716235 1.1750167 391.67359 1.0115970
## beta[2] 0.5990427 0.6554967  98.98279 1.0088166
## beta[3] 0.1044230 0.1093843 613.22428 0.9978156
## beta[4] 0.2064079 0.2182454 121.59685 1.0056875
## sigma   0.1623019 0.1646189 320.75100 1.0104805
```

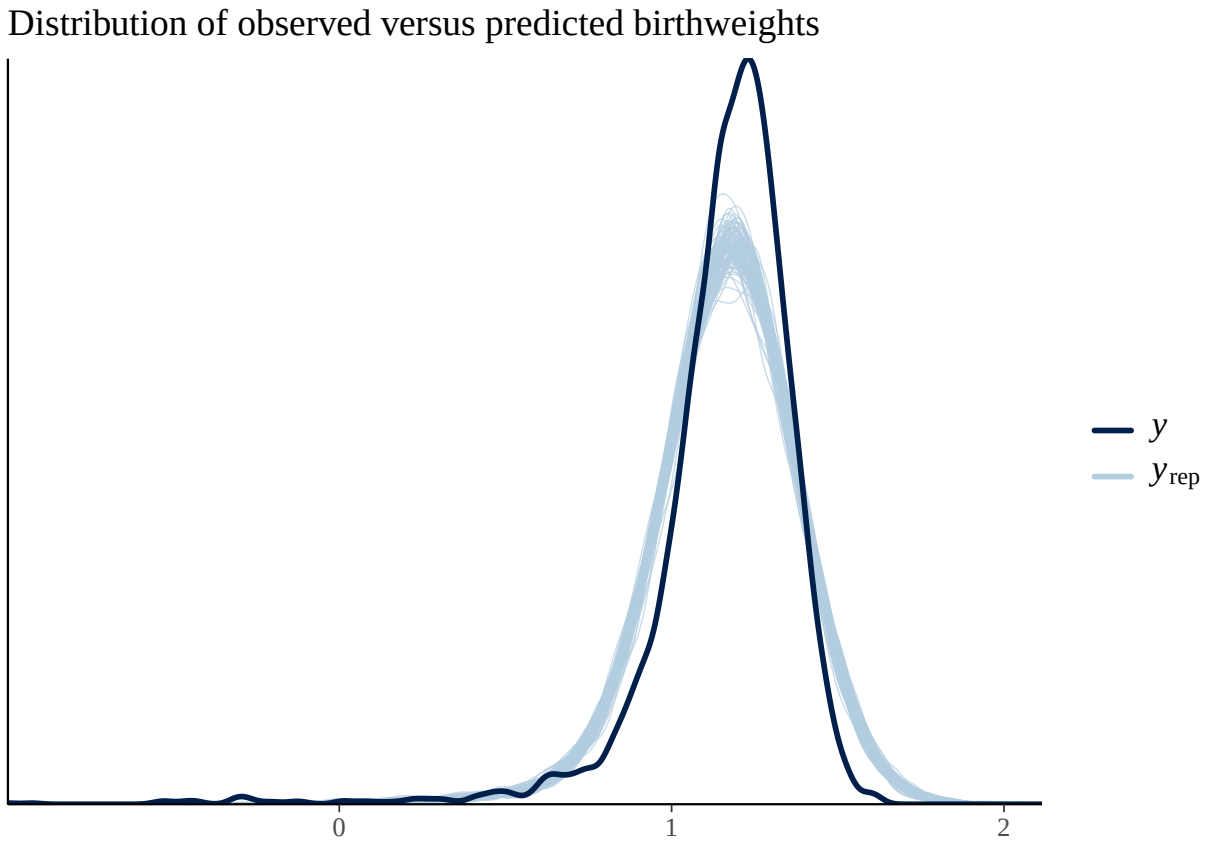
```
summary(mod2)$summary[c("beta[1]", "beta[2]", "beta[3]", "beta[4]", "sigma"),]
```

```
##              mean      se_mean          sd       2.5%        25%       50%
## beta[1] 1.1697241 1.385590e-04 0.002742186 1.16453578 1.16767109 1.1699278
## beta[2] 0.5563133 5.835253e-03 0.058054991 0.43745504 0.51708255 0.5561553
## beta[3] 0.1020960 1.481816e-04 0.003669476 0.09459462 0.09997153 0.1020339
## beta[4] 0.1967671 1.129799e-03 0.012458398 0.17164533 0.18817091 0.1974114
## sigma   0.1610727 9.950037e-05 0.001782004 0.15784213 0.15978020 0.1610734
##               75%     97.5%     n_eff      Rhat
## beta[1] 1.1716235 1.1750167 391.67359 1.0115970
## beta[2] 0.5990427 0.6554967  98.98279 1.0088166
## beta[3] 0.1044230 0.1093843 613.22428 0.9978156
## beta[4] 0.2064079 0.2182454 121.59685 1.0056875
## sigma   0.1623019 0.1646189 320.75100 1.0104805
```

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

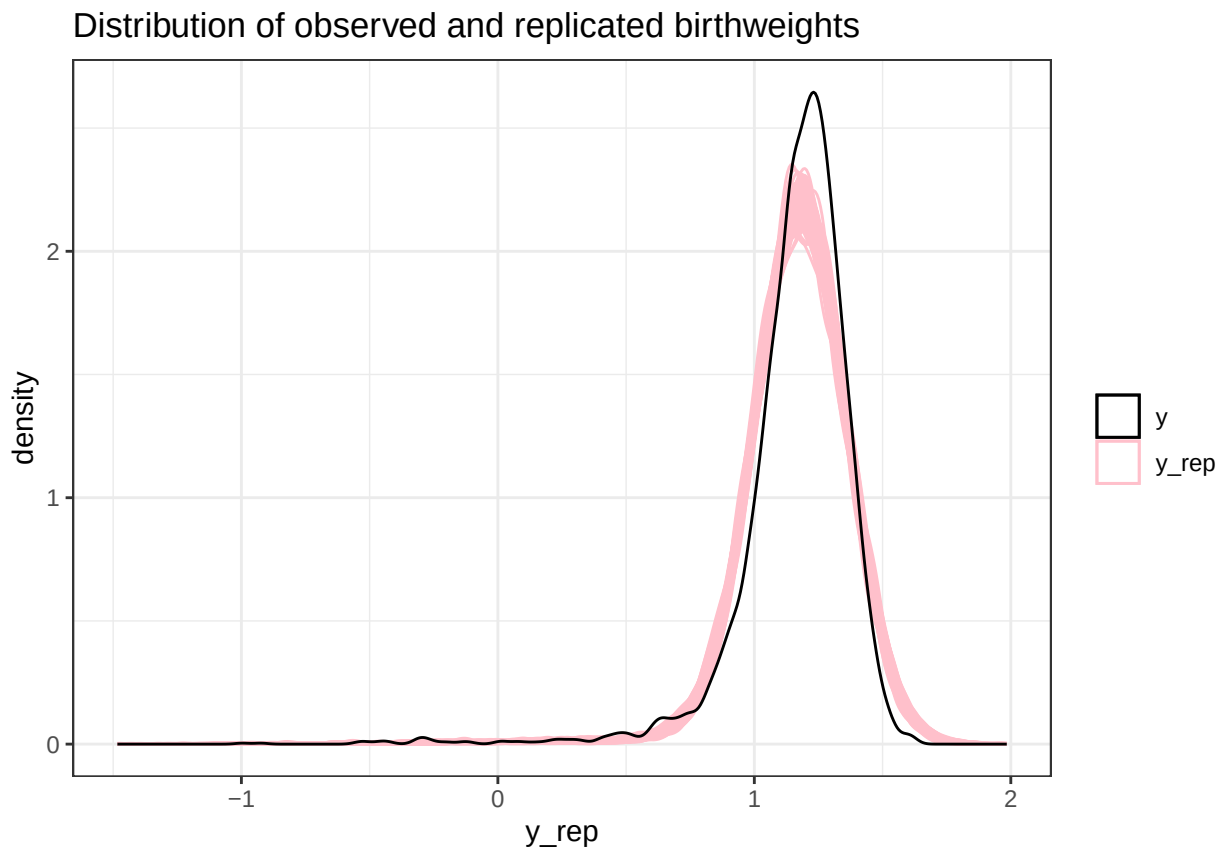


Question 6

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"]]
samp100 <- sample(nrow(yrep2), 100)
yrep2samp100 <- yrep2[samp100,]

# first, get into a tibble
rownames(yrep2) <- 1:nrow(yrep2)
dr <- as_tibble(t(yrep2))
dr <- dr %>% bind_cols(i = 1:ncol(yrep2), log_weight_obs = log(ds$birthweight))

# turn into long format; easier to plot
dr <- dr %>%
  pivot_longer(-(i:log_weight_obs), names_to = "sim", values_to ="y_rep")

# filter to just include 100 draws and plot!
dr %>%
  filter(sim %in% samp100) %>%
  ggplot(aes(y_rep, group = sim)) +
  geom_density(alpha = 0.2, aes(color = "y_rep")) +
```

```
  geom_density(data = ds %>% mutate(sim = 1),
               aes(x = log(birthweight), col = "y")) +
  scale_color_manual(name = "", values = c("y" = "black", "y_rep" = "pink")) +
  ggtitle("Distribution of observed and replicated birthweights") +
  theme_bw()
```

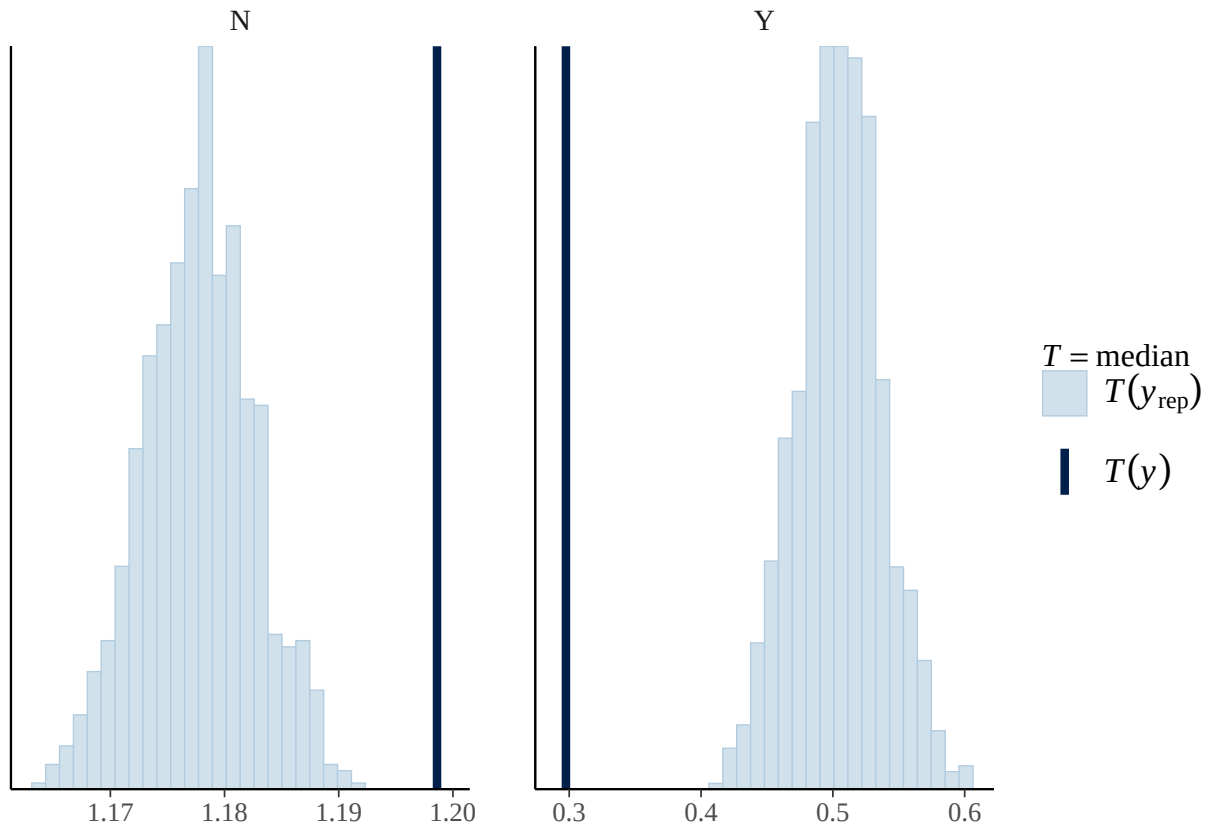


Test statistics

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

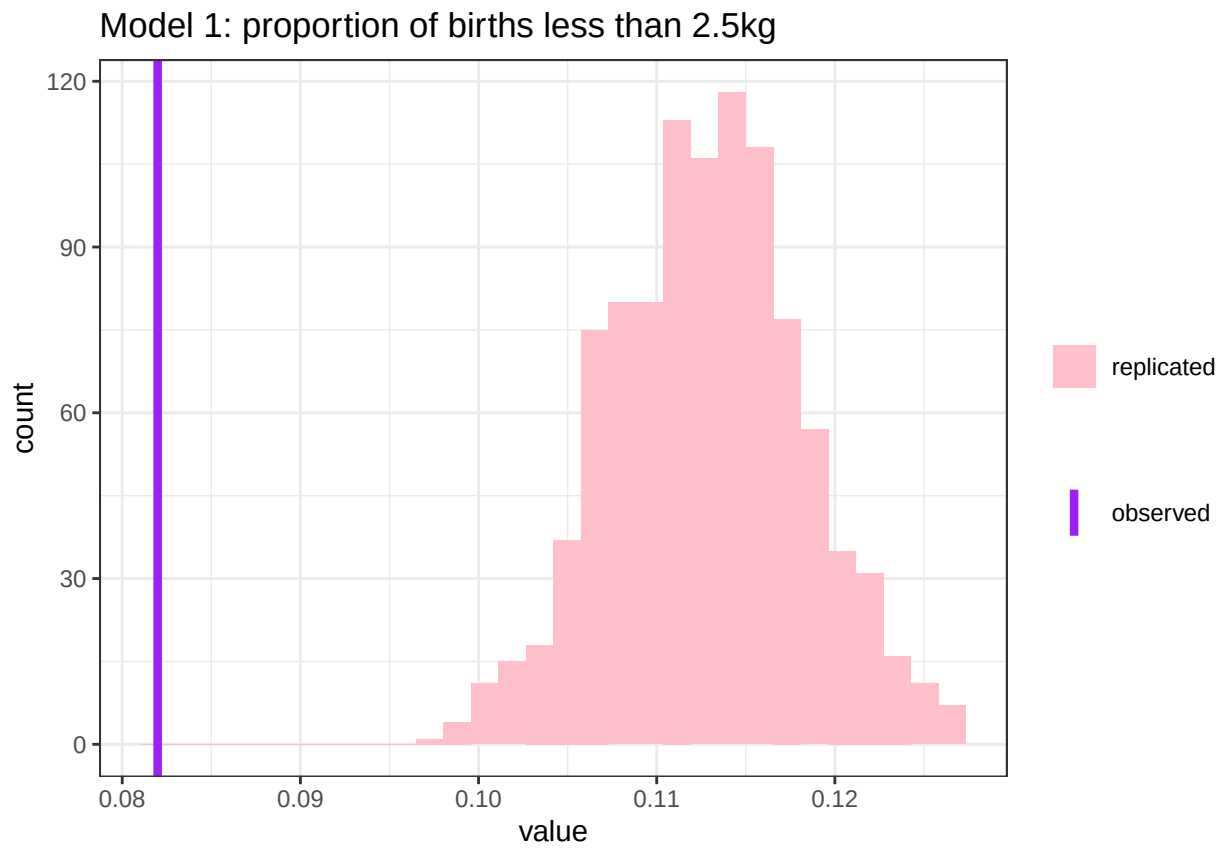
```
ppc_stat_grouped(ds$log_weight, yrep1, group = ds$preterm, stat = 'median')
```



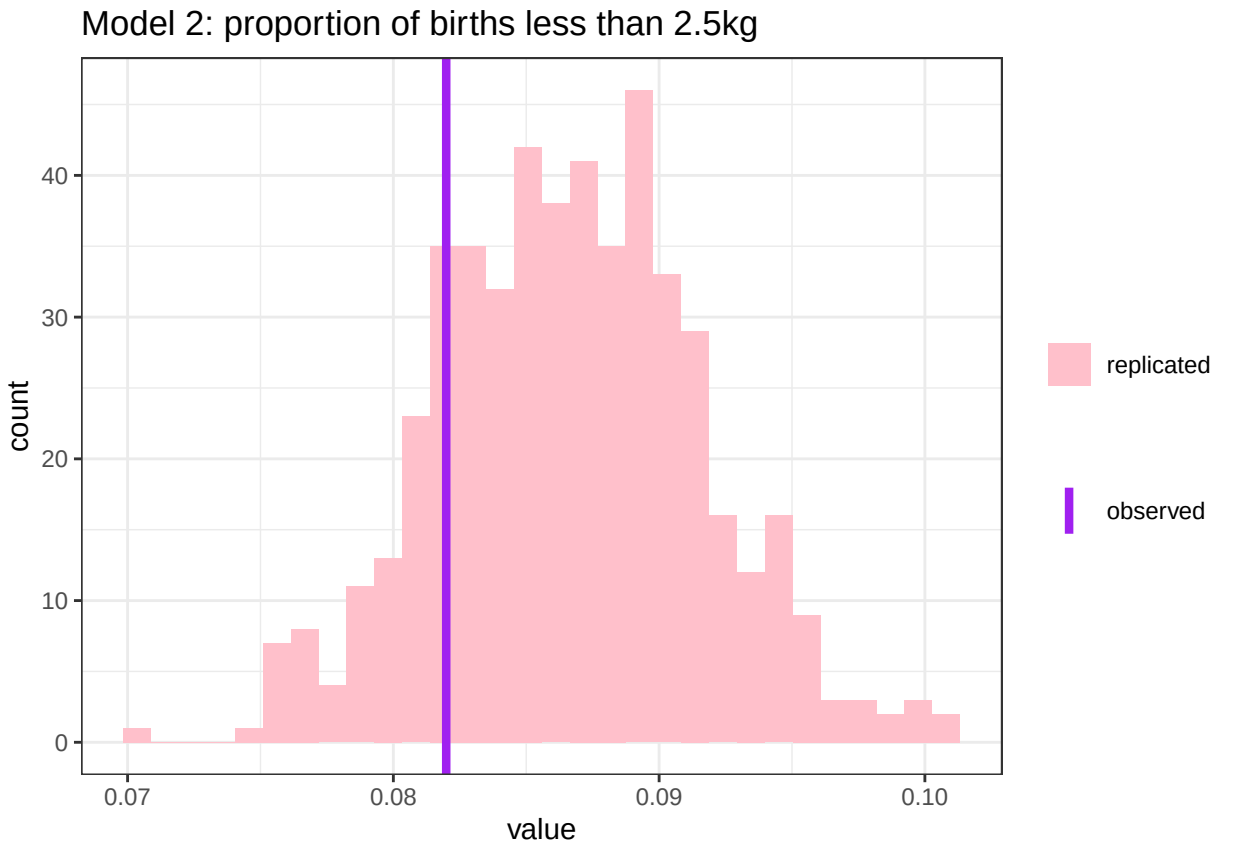
Question 7

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).

```
teststat <- mean(ds$log_weight<=log(2.5))

teststat_rep <- sapply(1:nrow(yrep1), function(i) mean(yrep1[i,]<=log(2.5)))
teststat_rep_2 <- sapply(1:nrow(yrep2), function(i) mean(yrep2[i,]<=log(2.5)))

ggplot(data = as_tibble(teststat_rep), aes(value)) +
    geom_histogram(aes(fill = "replicated")) +
    geom_vline(aes(xintercept = teststat, color = "observed"), lwd = 1.5) +
  ggtitle("Model 1: proportion of births less than 2.5kg") +
  scale_color_manual(name = "", values = c("observed" = "purple"))+
  scale_fill_manual(name = "", values = c("replicated" = "pink")) + theme_bw()
```




```
ggplot(data = as_tibble(teststat_rep_2), aes(value)) +
    geom_histogram(aes(fill = "replicated")) +
    geom_vline(aes(xintercept = teststat, color = "observed"), lwd = 1.5) +
  ggtitle("Model 2: proportion of births less than 2.5kg") +
  scale_color_manual(name = "", values = c("observed" = "purple"))+
  scale_fill_manual(name = "", values = c("replicated" = "pink")) + theme_bw()
```



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   1377.0  72.4
```

```
## p_loo         9.8   1.4
## looic     -2754.1 144.8
## ------
## 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 500 by 3842 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo   1552.8  70.0
## p_loo        14.8   2.3
## looic     -3105.6 139.9
## ------
## Monte Carlo SE of elpd_loo is 0.2.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

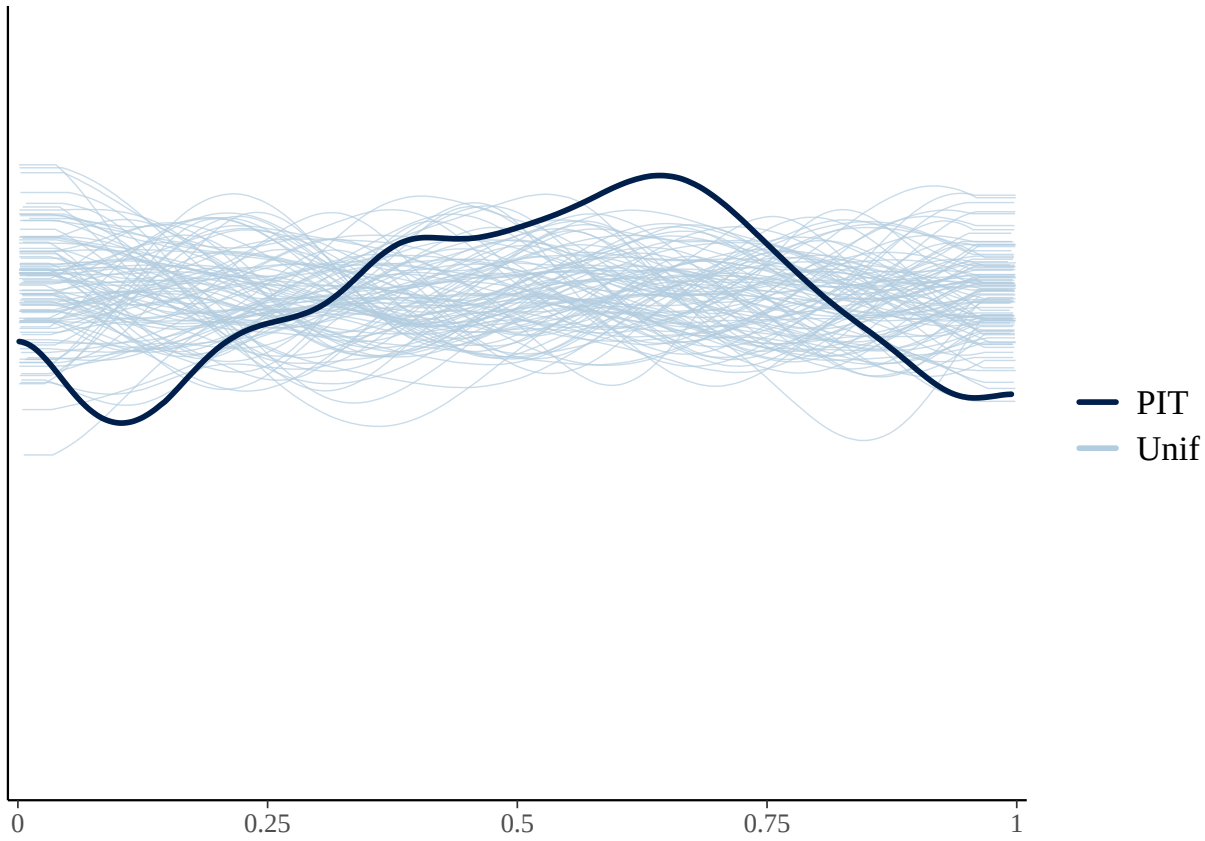
Comparing the two models tells us Model 2 is better:

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

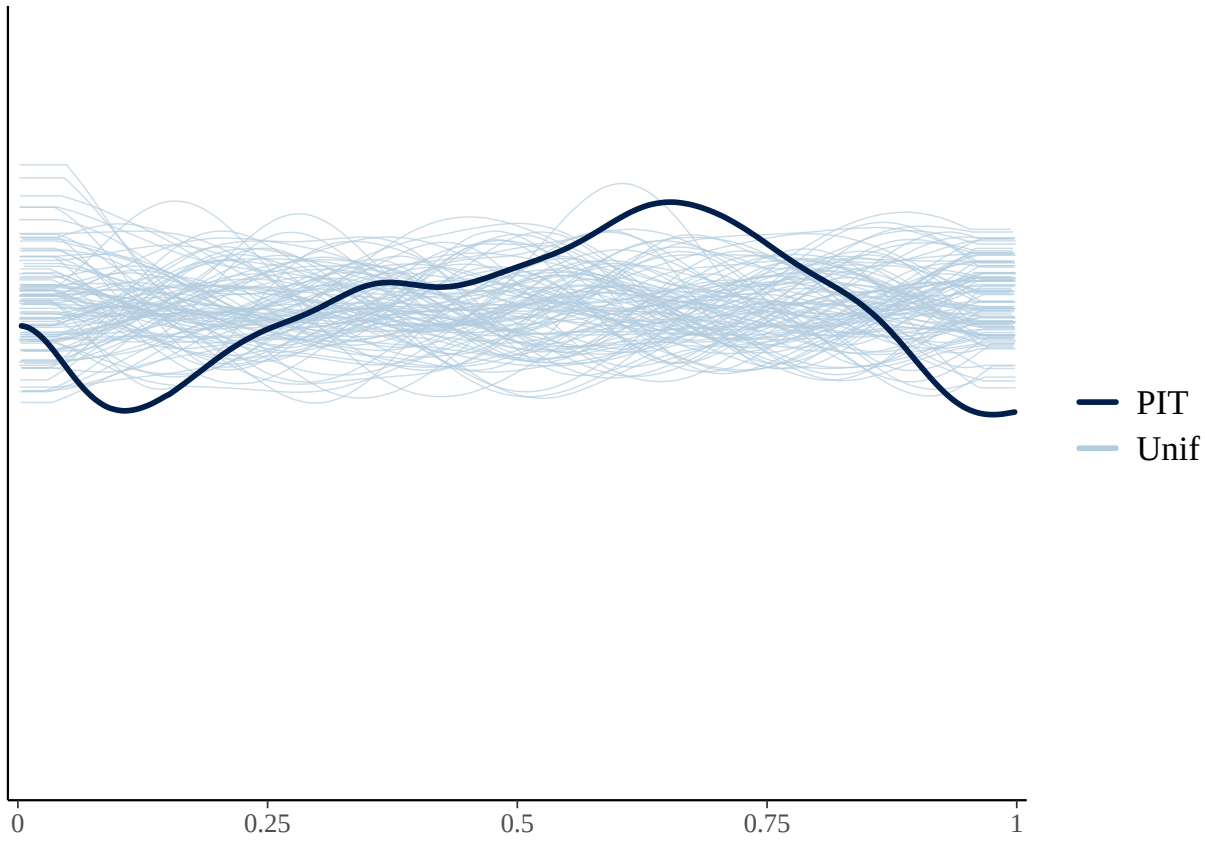
```
##        elpd_diff se_diff
## model2    0.0       0.0
## model1 -175.8      36.2
```

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))
```




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



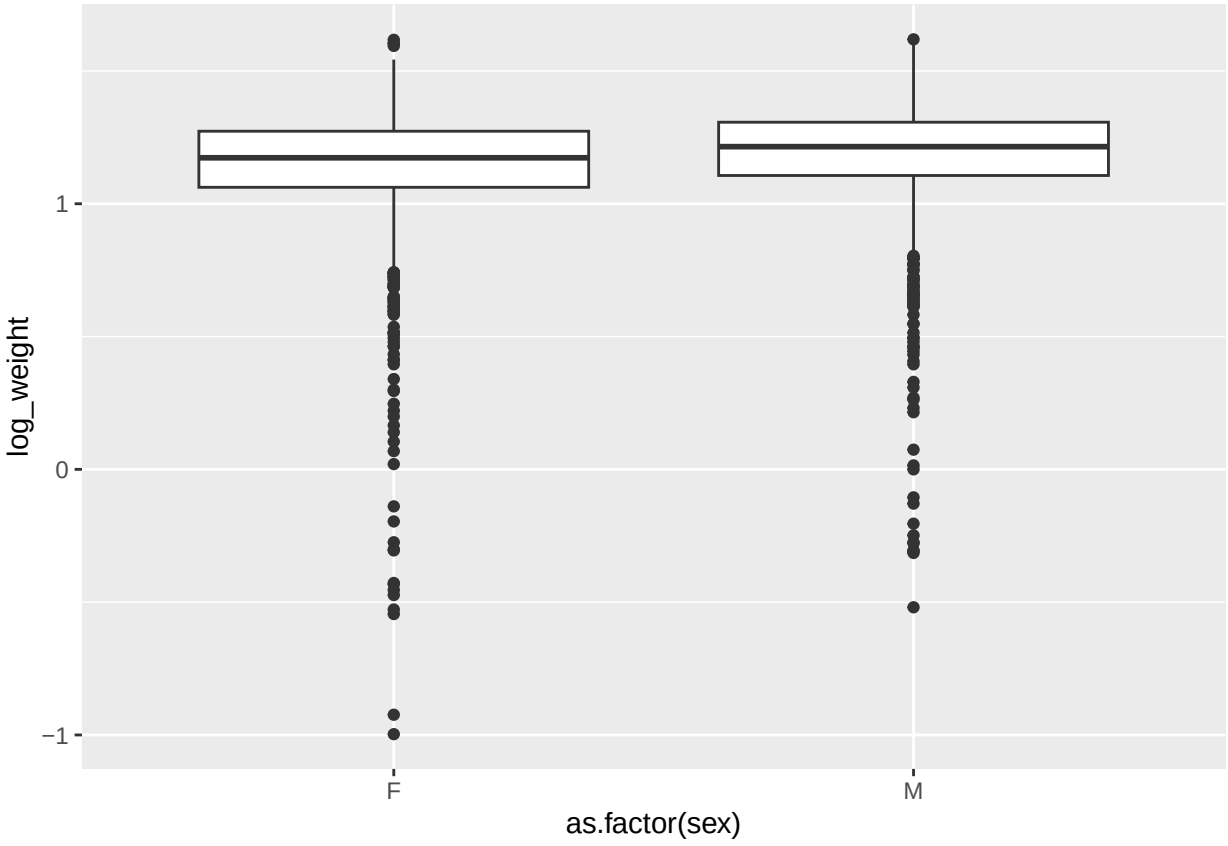
Bonus question (not required)

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

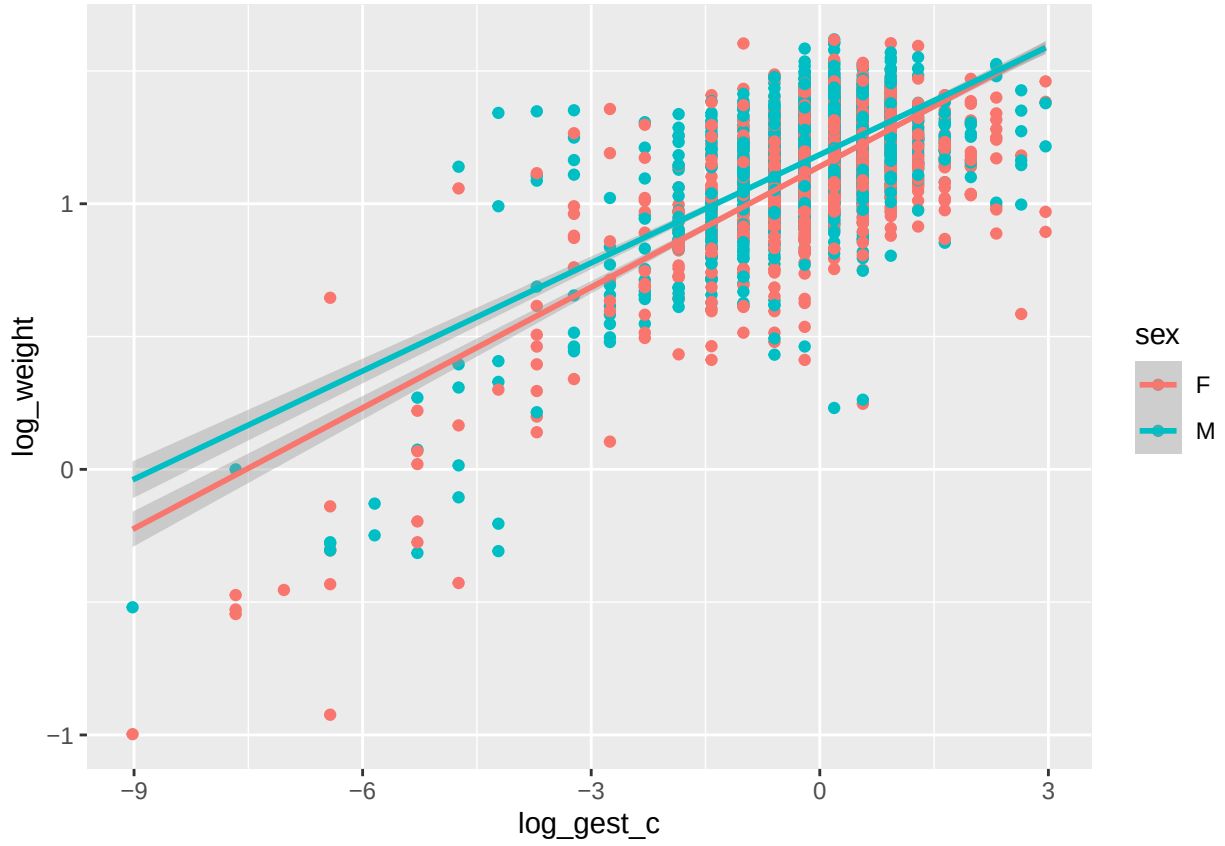
Question 8

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.

```
ggplot(ds, aes(x=as.factor(sex), y=log_weight)) + geom_boxplot()
```




```
ds %>% ggplot(aes(x=log_gest_c, y=log_weight, color=sex)) + geom_point() + geom_smooth(method="lm")
```



We add sex to Model 2.

```
ds %>% mutate(sex_num = ifelse(sex == "F", 0, 1)) -> ds

# put into a list
stan_data3 <- list(N = nrow(ds),
                   log_weight = ds$log_weight,
                   log_gest = ds$log_gest_c,
                   preterm = ds$preterm_f,
                   log_gest_preterm = ds$log_gest_preterm,
                   sex = ds$sex_num)

mod3 <- stan(data = stan_data3,
             file = here("simple_weight3.stan"),
             iter = 500,
             seed = 243)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.016163 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 161.63 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 500 [  0%]  (Warmup)
```

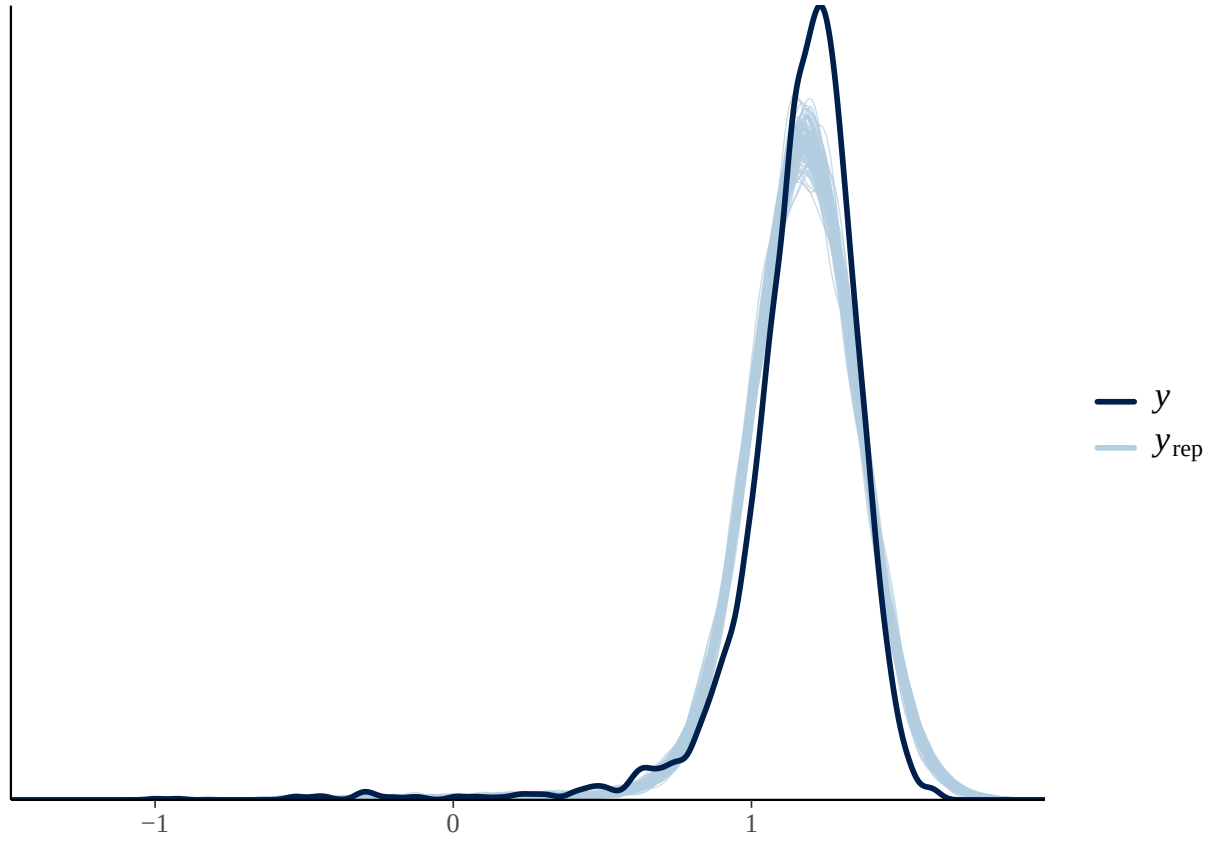
```
## Chain 1: Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 1: Iteration: 100 / 500 [ 20%]  (Warmup)
## Chain 1: Iteration: 150 / 500 [ 30%]  (Warmup)
## Chain 1: Iteration: 200 / 500 [ 40%]  (Warmup)
## Chain 1: Iteration: 250 / 500 [ 50%]  (Warmup)
## Chain 1: Iteration: 251 / 500 [ 50%]  (Sampling)
## 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: 5.142 seconds (Warm-up)
## Chain 1:                6.144 seconds (Sampling)
## Chain 1:                11.286 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000875 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 8.75 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: 4.707 seconds (Warm-up)
## Chain 2:                3.966 seconds (Sampling)
## Chain 2:                8.673 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.001084 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 10.84 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: 5.443 seconds (Warm-up)
## Chain 3:                4.263 seconds (Sampling)
## Chain 3:                9.706 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000885 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 8.85 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: 6.399 seconds (Warm-up)
## Chain 4:                3.595 seconds (Sampling)
## Chain 4:                9.994 seconds (Total)
## Chain 4:
```

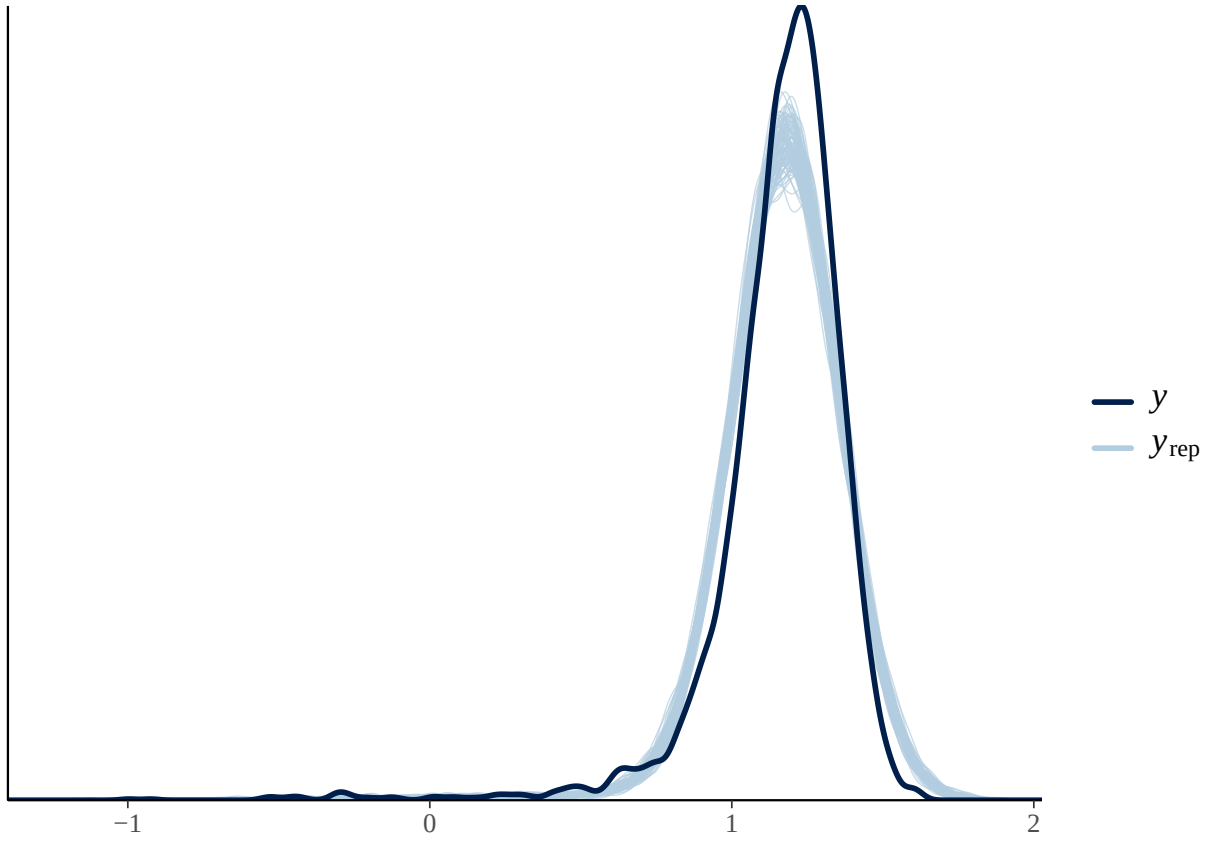
```
summary(mod3)$summary[c("beta[1]", "beta[2]", "beta[3]", "beta[4]", "beta[5]", "sigma"),]
```

```
##               mean      se_mean          sd       2.5%       25%        50%
## beta[1] 1.14858743 1.255982e-04 0.003865342 1.14095280 1.1460639 1.14862377
## beta[2] 0.10239565 9.937416e-05 0.003570133 0.09524595 0.1000605 0.10243417
## beta[3] 0.55089942 3.794993e-03 0.067149999 0.42434578 0.5073889 0.54944835
## beta[4] 0.19561626 7.897026e-04 0.013944263 0.16899315 0.1861179 0.19547056
## beta[5] 0.04208808 1.867449e-04 0.005563357 0.03113255 0.0383279 0.04186556
## sigma   0.15987251 6.591160e-05 0.001801103 0.15662063 0.1586474 0.15980923
##                75%      97.5%     n_eff      Rhat
## beta[1] 1.15099873 1.15621903  947.1289 0.9988379
## beta[2] 0.10484552 0.10936426 1290.6902 0.9986903
## beta[3] 0.59404976 0.69013748  313.0907 1.0055996
## beta[4] 0.20418632 0.22269684  311.7913 1.0066296
## beta[5] 0.04561499 0.05322607  887.5159 0.9972294
## sigma   0.16111856 0.16344070  746.7125 1.0143192
```

```
yrep2 <- extract(mod2)[["log_weight_rep"]]
ppc_dens_overlay(y, yrep2[samp100, ])
```

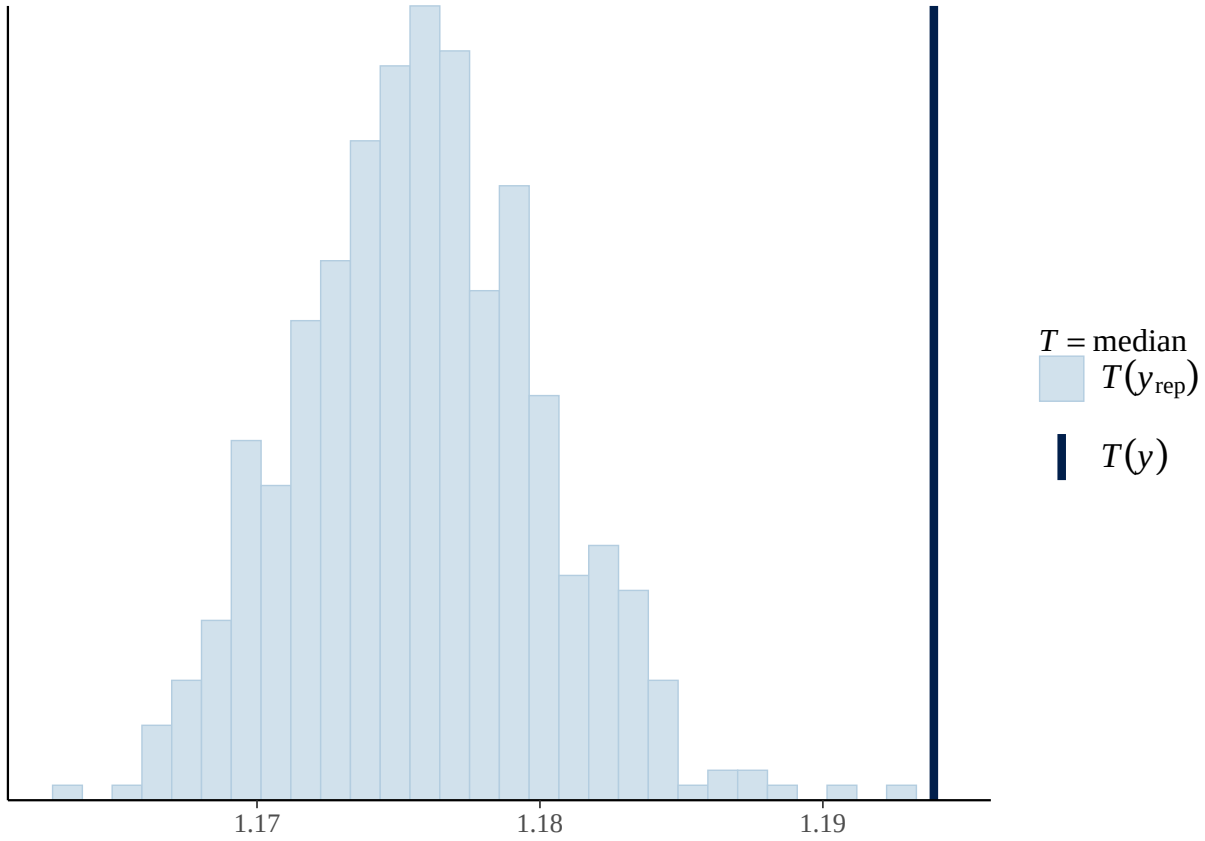



```
yrep3 <- extract(mod3)[["log_weight_rep"]]
ppc_dens_overlay(y, yrep3[samp100, ])
```

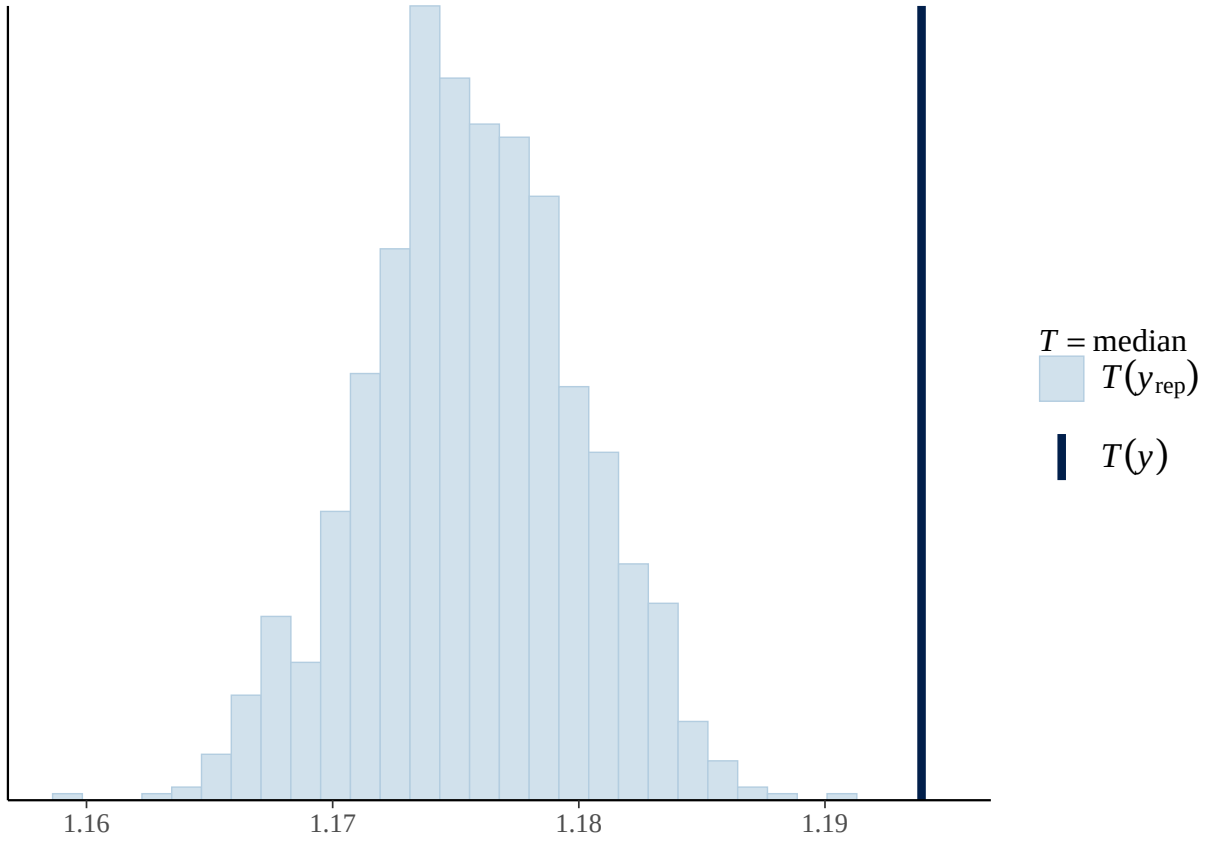


Model 2 seems to be marginally better though the difference is not very significant.

```
ppc_stat(ds$log_weight, yrep2, stat = 'median')
```




```
ppc_stat(ds$log_weight, yrep3, stat = 'median')
```



Model 3 seems to be slightly better.