Week 6: Visualizing the Bayesian Workflow

21/02/23

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)
```

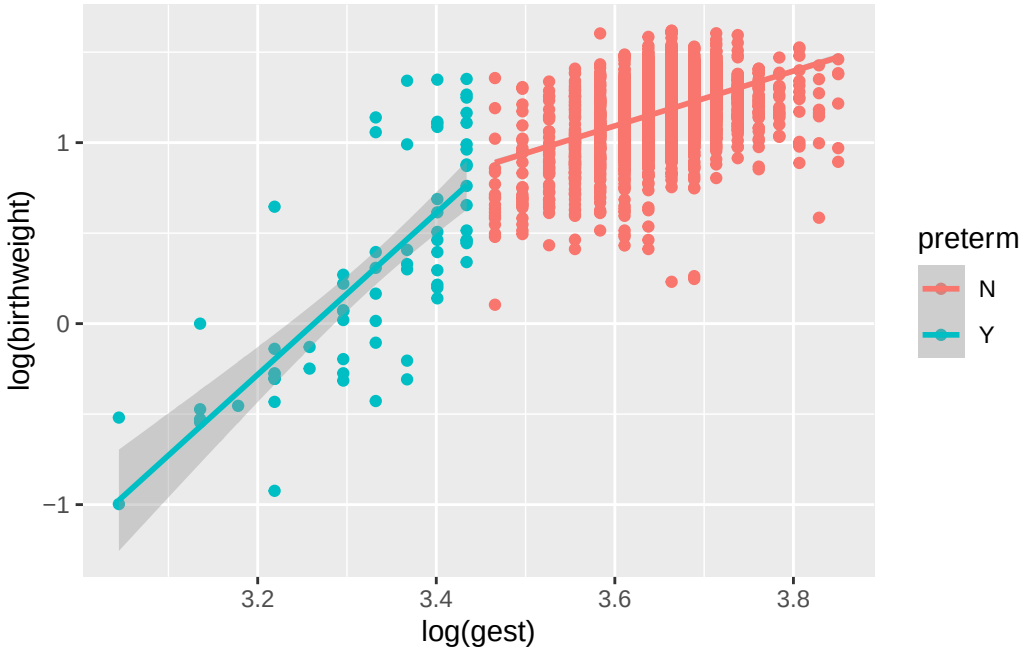
Question 1

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

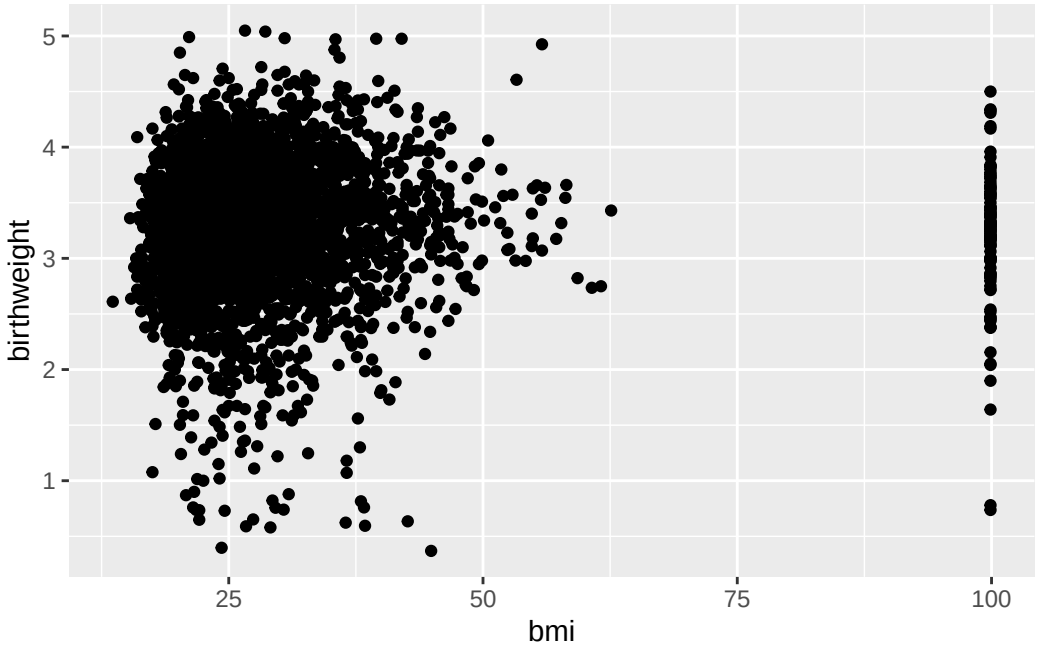
Feel free to replicate one of the scatter plots in the lectures as one of the interesting observations, as those form the basis of our models.

```
ds |>
  ggplot(aes(x=log(gest), y=log(birthweight), color=preterm)) +
  geom_point()+
  geom_smooth(method = "lm")
```



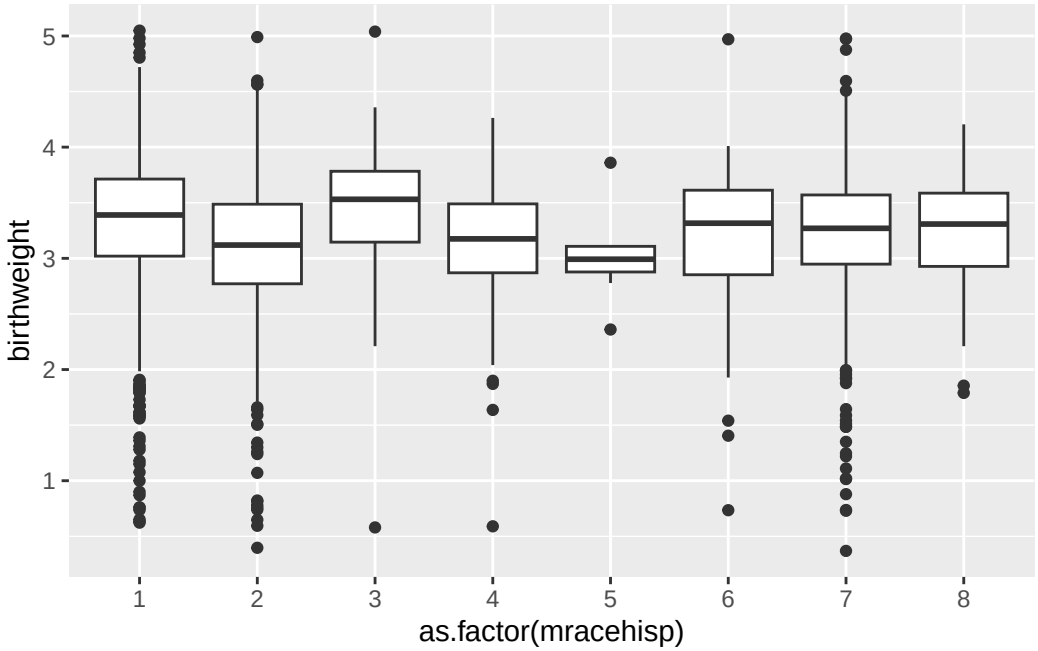
I plotted the scatter plot of log(birthweight) vs log(gestational age) which forms the basis of the model. I also use different color scales to indicate whether the birth was preterm or not. I found that log(birthweight) increases as log(gestational age) increases, and the increase is faster (slope is steeper) in preterm birth.

```
ds |>
  ggplot(aes(x=bmi, y=birthweight)) +
  geom_point()
```



I plotted a scatter plot of birthweight vs bmi. Overall, there is no clear pattern between these two variables. On a side note, a bmi value close to 100 seems to be an outlier.

```
ds |>
  ggplot(aes(x=as.factor(mracehisp), y=birthweight))+
  geom_boxplot()
```



I plotted boxplots of birthweight by mom's race, and found that race 5 (Non-Hispanic NHOPI (only)) has denser concentration around birthweight 3kg while other races' birthweight are more spread out. Race 1 (Non-Hispanic White (only)), 2 (Non-Hispanic Black (only)) and 7 (Hispanic) have more outlier birthweights than other races.

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).

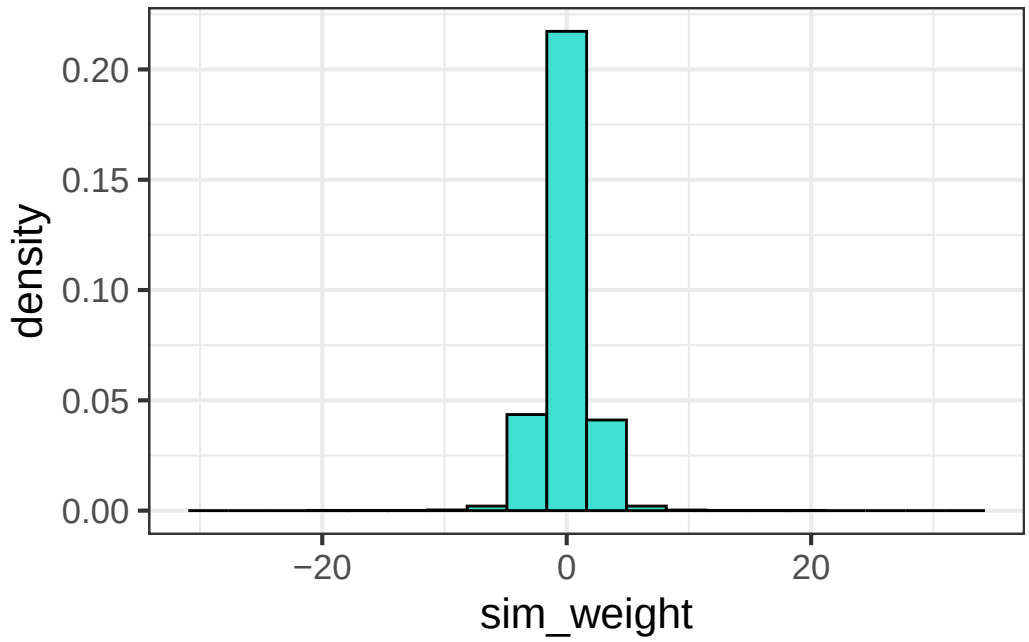
Question 2

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**.

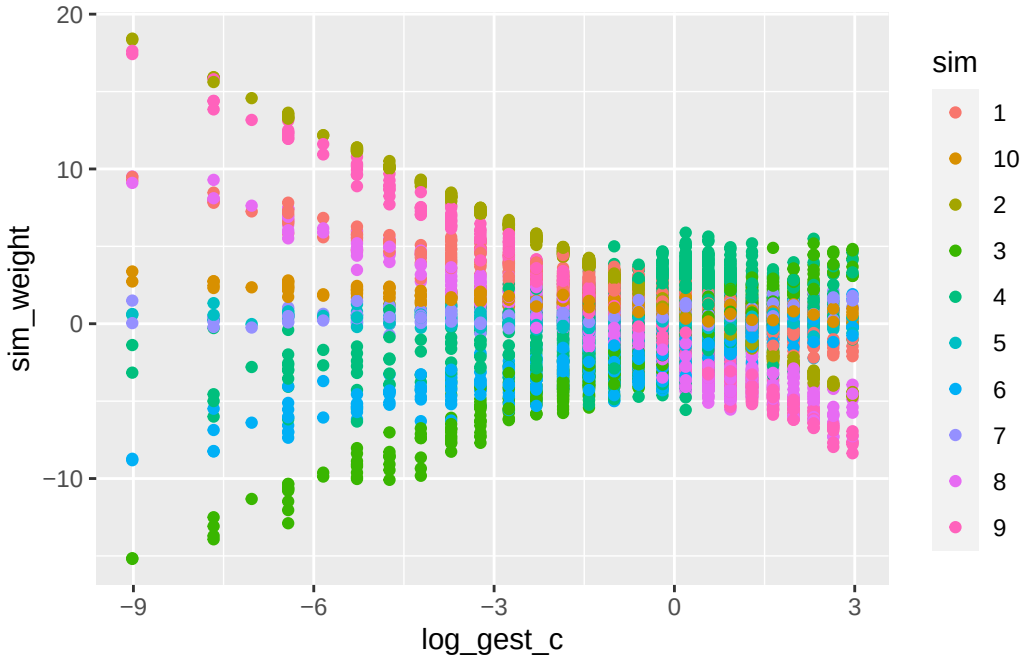
- Plot the resulting distribution of simulated (log) birth weights.
- Plot ten simulations of (log) birthweights against gestational age.

```
# code inspired from https://www.monicaalexander.com/posts/2020-28-02-bayes_viz/
set.seed(1)
nsims=1000
sigma <- abs(rnorm(nsims, 0, 1))
beta1 <- rnorm(nsims, 0, 1)
beta2 <- rnorm(nsims, 0, 1)

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

for(i in 1:nsims){
  this_mu <- beta1[i] + beta2[i]*dsims$log_gest_c
  dsims[paste0(i)] <- this_mu + rnorm(nrow(dsims), 0, sigma[i])
}

dsl <- dsims %>%
  pivot_longer(`1`:`1000`, names_to = "sim", values_to = "sim_weight")
```

```r
dsl %>%
  ggplot(aes(sim_weight)) + geom_histogram(aes(y = ..density..), bins = 20, fill = "turquo
  theme_bw(base_size = 16)
```




```r
dsl_10 <- dsims[,1:11] |>
  pivot_longer(`1`:`10`, names_to = "sim", values_to = "sim_weight")

dsl_10|>
  ggplot(aes(x=log_gest_c, y=sim_weight,color=sim))+
  geom_point()
```



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 -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEI
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeac
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEige
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEige
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Co
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Co
namespace Eigen {
              ^
              ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeac
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEige
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/Core:9
#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 0.000511 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 5.11 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: 0.695376 seconds (Warm-up)
```

```
Chain 1:                0.555321 seconds (Sampling)
Chain 1:                1.2507 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'simple_weight' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000233 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.33 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: 0.611782 seconds (Warm-up)
Chain 2:                0.571752 seconds (Sampling)
Chain 2:                1.18353 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'simple_weight' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000249 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.49 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: 0.633558 seconds (Warm-up)
Chain 3:                0.563526 seconds (Sampling)
Chain 3:                1.19708 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'simple_weight' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000252 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.52 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: 0.623472 seconds (Warm-up)
Chain 4:                0.578069 seconds (Sampling)
Chain 4:                1.20154 seconds (Total)
Chain 4:
```

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

```
             mean      se_mean          sd      2.5%       25%       50%
beta[1] 1.1626250 7.634607e-05 0.002583881 1.1575321 1.1609497 1.1626383
beta[2] 0.1436183 8.105504e-05 0.002791943 0.1380281 0.1417563 0.1436199
```

```
sigma   0.1689127 1.051837e-04 0.001979909 0.1650908 0.1676042 0.1688619
              75%     97.5%     n_eff      Rhat
beta[1] 1.1643919 1.1677313 1145.4383 0.9970543
beta[2] 0.1455075 0.1489575 1186.4598 0.9984953
sigma   0.1701148 0.1728405  354.3181 1.0046933
```

Question 3

Based on model 3, give an estimate of the expected birthweight of a baby who was born at a gestational age of 37 weeks.

```
exp(1.1626250+0.1436183*(log(37) - mean(log(ds$gest)))/sd(log(ds$gest)))
```

```
[1] 2.93654
```

The expected birthweight is 2.93654 kg.

Question 4

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

```
ds$log_weight <- log(ds$birthweight)
ds$log_gest_c <- (log(ds$gest) - mean(log(ds$gest)))/sd(log(ds$gest))
ds$prematurity <- ifelse(ds$preterm=="Y", 1, 0)
# put into a list
stan_data <- list(N = nrow(ds),
                  log_weight = ds$log_weight,
                  log_gest = ds$log_gest_c,
                  prematurity = ds$prematurity,
                  combo = ds$prematurity*ds$log_gest_c)
```

```
# model is beta1+beta3*log_gest+beta2*premature+beta4*log_gest*premature
mod2 <- stan(data = stan_data,
             file = here("code/models/simple_weight_mod2.stan"),
             iter = 500,
             seed = 243)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEI
```

```
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHea
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEig
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEig
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/C
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/C
namespace Eigen {
              ^
              ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHea
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEig
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/Core:
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'simple_weight_mod2' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.00166 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 16.6 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: 2.92786 seconds (Warm-up)
Chain 1:                2.47478 seconds (Sampling)
Chain 1:                5.40264 seconds (Total)
```

```
Chain 1:

SAMPLING FOR MODEL 'simple_weight_mod2' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000647 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 6.47 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: 3.25294 seconds (Warm-up)
Chain 2:                2.4165 seconds (Sampling)
Chain 2:                5.66944 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'simple_weight_mod2' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000638 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 6.38 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: 3.17071 seconds (Warm-up)
Chain 3:                2.44085 seconds (Sampling)
Chain 3:                5.61156 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'simple_weight_mod2' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000619 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 6.19 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: 2.65941 seconds (Warm-up)
Chain 4:                2.65094 seconds (Sampling)
Chain 4:                5.31035 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] 1.1695756 8.547704e-05 0.002616101 1.16438077 1.16777524 1.1695827
beta[2] 0.5600955 3.907742e-03 0.061471283 0.43194048 0.51978992 0.5628837
beta[3] 0.1017901 1.122621e-04 0.003472556 0.09496487 0.09953192 0.1017907
beta[4] 0.1980772 8.197160e-04 0.012882569 0.17057835 0.18906497 0.1985114
```

```
sigma   0.1612549 8.293335e-05 0.001879479 0.15782955 0.16002138 0.1612254
              75%     97.5%    n_eff      Rhat
beta[1] 1.1714038 1.1745871 936.7202 0.9994706
beta[2] 0.6036808 0.6706357 247.4531 1.0094054
beta[3] 0.1040705 0.1088809 956.8241 0.9995333
beta[4] 0.2077460 0.2220285 246.9893 1.0117629
sigma   0.1625277 0.1649652 513.5902 1.0058834
```

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"),]
```

```
             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

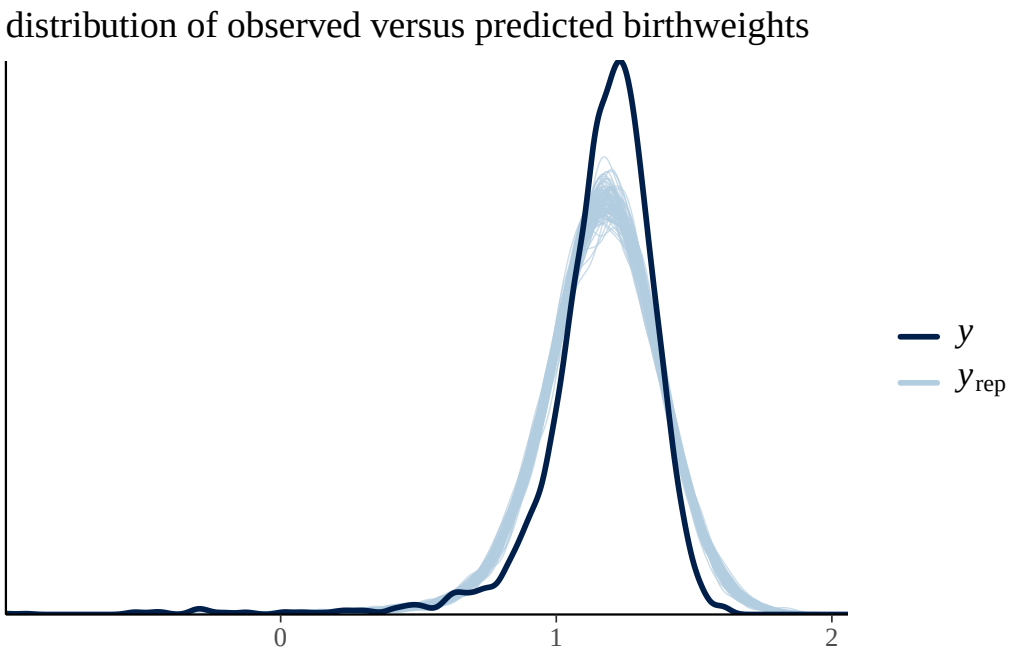
Now we've run two candidate models let's do some posterior predictive checks. The `bayesplot` package has a lot of inbuilt graphing functions to do this. For example, let's plot the distribution of our data (y) against 100 different datasets drawn from the posterior predictive distribution:

```
set.seed(1856)
y <- ds$log_weight
yrep1 <- extract(mod1)[["log_weight_rep"]]
yrep2 <- extract(mod2)[["log_weight_rep"]]
```

```
dim(yrep1)
```

```
[1] 1000 3842
```

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

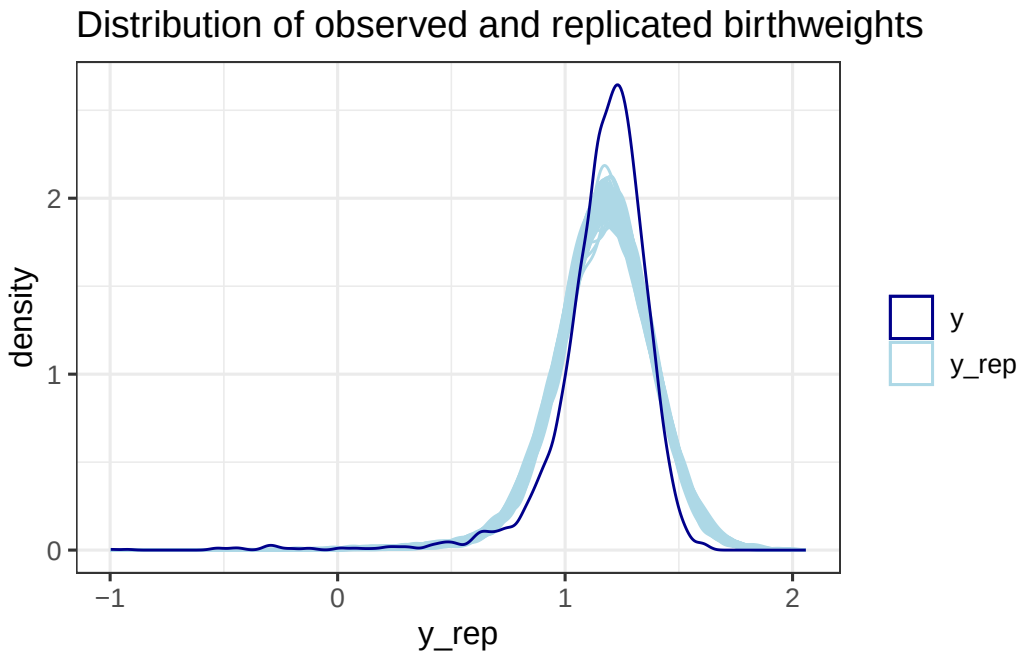


Question 6

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`)

```
# code inspired from https://www.monicaalexander.com/posts/2020-28-02-bayes_viz/
# first, get into a tibble
rownames(yrep1) <- 1:nrow(yrep1)
dr <- as_tibble(t(yrep1))
dr <- dr %>% bind_cols(i = 1:length(ds$birthweight), log_weight_obs = log(ds$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" = "darkblue",
                                "y_rep" = "lightblue")) +
  ggtitle("Distribution of observed and replicated birthweights") +
  theme_bw(base_size = 12)
```

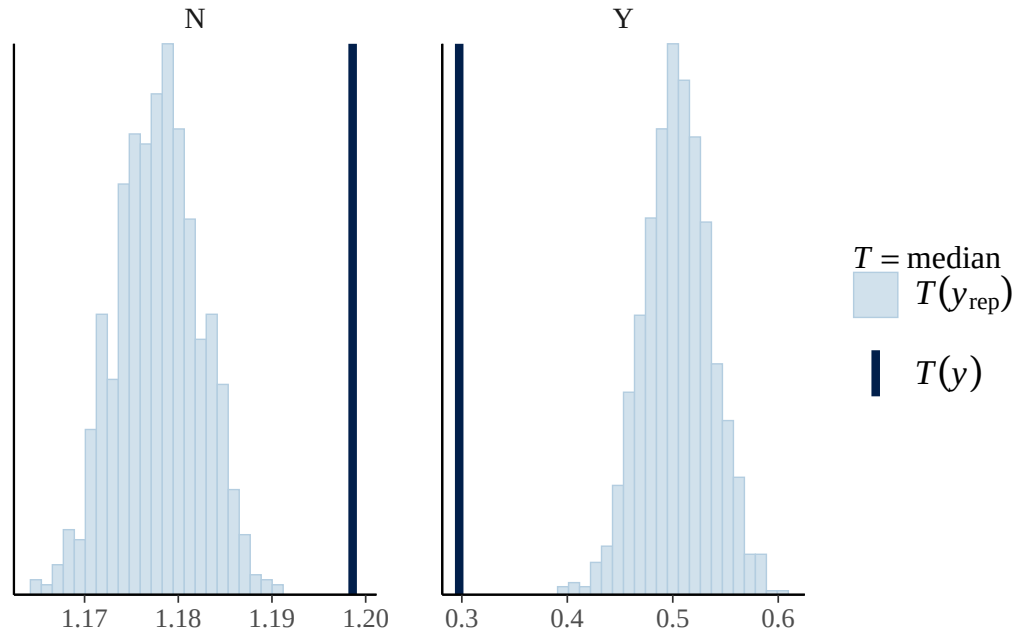


Test statistics

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')
```

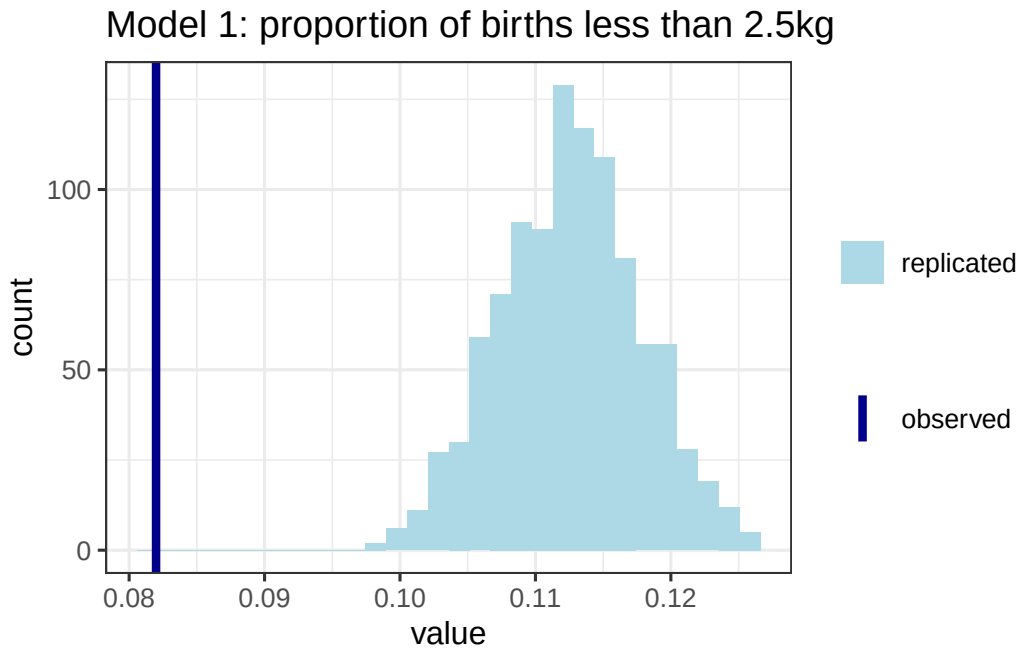


Question 7

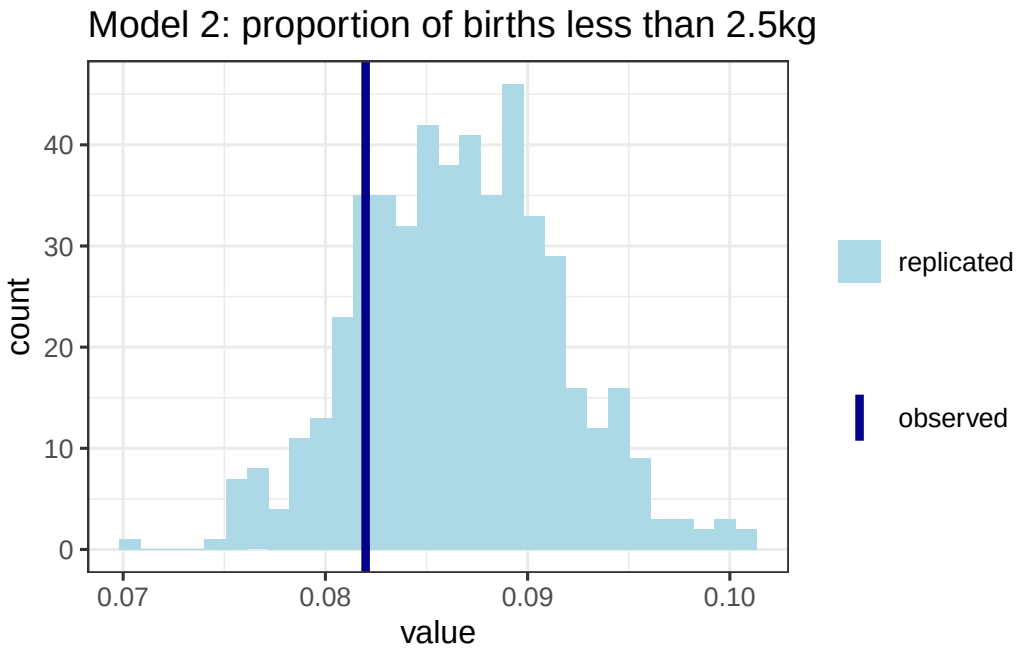
Use a test statistic of the proportion of births under 2.5kg. Calculate the test statistic for the data, and the posterior predictive samples for both models, and plot the comparison (one plot per model).

```
# code inspired from https://www.monicaalexander.com/posts/2020-28-02-bayes_viz/
t_y <- mean(y<=log(2.5))
t_y_rep <- sapply(1:nrow(yrep1), function(i) mean(yrep1[i,]<=log(2.5)))
t_y_rep_2 <- sapply(1:nrow(yrep2), function(i) mean(yrep2[i,]<=log(2.5)))

ggplot(data = as_tibble(t_y_rep), aes(value)) +
    geom_histogram(aes(fill = "replicated")) +
    geom_vline(aes(xintercept = t_y, color = "observed"), lwd = 1.5) +
  ggtitle("Model 1: proportion of births less than 2.5kg") +
  theme_bw(base_size = 12) +
  scale_color_manual(name = "",
                     values = c("observed" = "darkblue"))+
  scale_fill_manual(name = "",
```

```
                    values = c("replicated" = "lightblue"))
```




```
ggplot(data = as_tibble(t_y_rep_2), aes(value)) +
    geom_histogram(aes(fill = "replicated")) +
    geom_vline(aes(xintercept = t_y, color = "observed"), lwd = 1.5) +
  ggtitle("Model 2: proportion of births less than 2.5kg") +
  theme_bw(base_size = 12) +
  scale_color_manual(name = "",
                     values = c("observed" = "darkblue"))+
  scale_fill_manual(name = "",
                     values = c("replicated" = "lightblue"))
```




```
t_y
```

```
[1] 0.08198855
```

The average of the proportion of births under 2.5kg is 0.08198855.

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   1377.2  72.6
p_loo         9.6   1.5
looic     -2754.5 145.2
------
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 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.
```

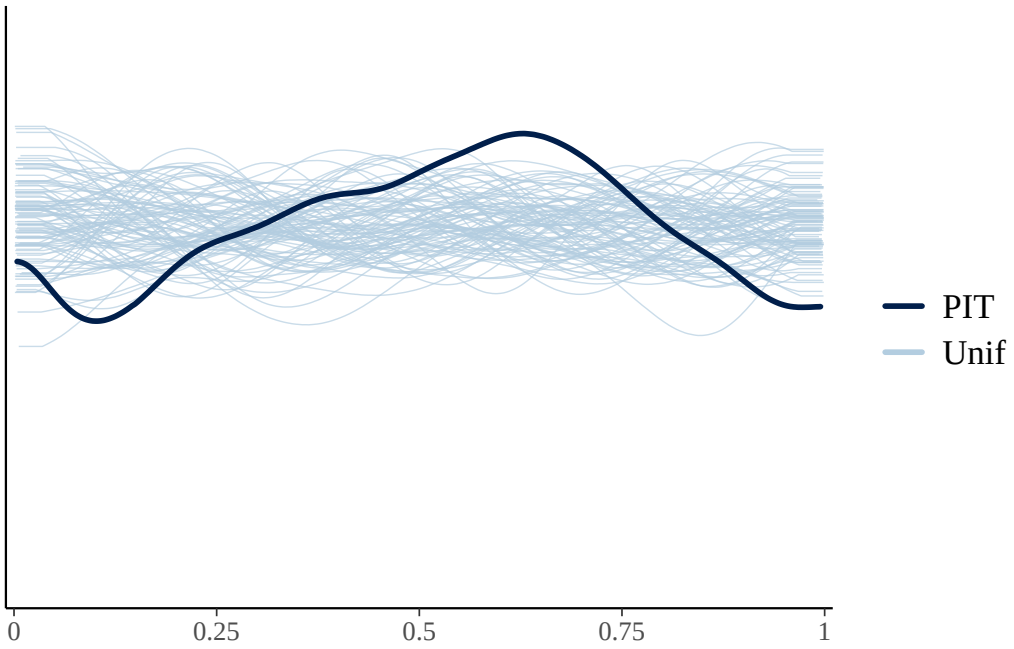
Comparing the two models tells us Model 2 is better:

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

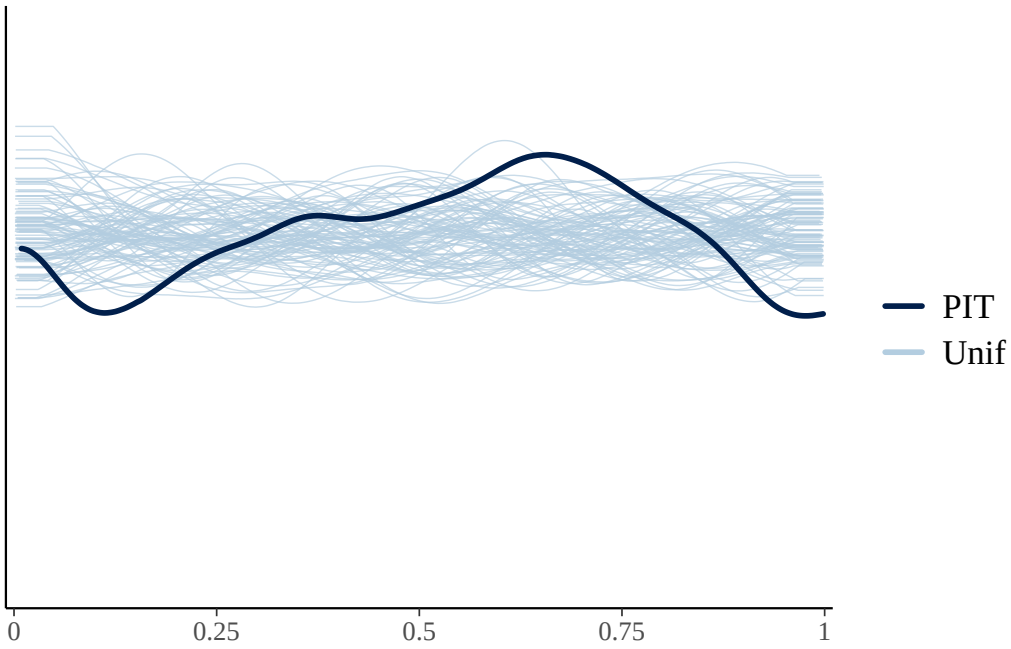
```
       elpd_diff se_diff
model2    0.0       0.0
model1 -175.6      36.3
```

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))
```




```
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.

I add (centered and standardized) (log) bmi to model2.
The model is beta1+beta3*log_gest+beta2*premature+beta4*log_gest*premature+beta5*log_bmi.

```
ds$log_weight <- log(ds$birthweight)
ds$log_gest_c <- (log(ds$gest) - mean(log(ds$gest)))/sd(log(ds$gest))
ds$prematurity <- ifelse(ds$preterm=="Y", 1, 0)
ds$log_bmi_c <- (log(ds$bmi) - mean(log(ds$bmi)))/sd(log(ds$bmi))
# put into a list
stan_data <- list(N = nrow(ds),
                  log_weight = ds$log_weight,
                  log_gest = ds$log_gest_c,
```

```
                  prematurity = ds$prematurity,
                  combo = ds$prematurity*ds$log_gest_c,
                  log_bmi = ds$log_bmi_c)
```

```
mod3 <- stan(data = stan_data,
             file = here("code/models/simple_weight_mod_q8.stan"),
             iter = 500,
             seed = 243)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEI
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeac
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEige
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEige
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Co
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Co
namespace Eigen {
              ^
              ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeac
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEige
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/Core:9
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'simple_weight_mod_q8' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.002096 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 20.96 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: 3.68258 seconds (Warm-up)
Chain 1:                2.70477 seconds (Sampling)
Chain 1:                6.38735 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'simple_weight_mod_q8' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000809 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 8.09 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: 4.05092 seconds (Warm-up)
Chain 2:                3.307 seconds (Sampling)
Chain 2:                7.35792 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'simple_weight_mod_q8' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000817 seconds
```

```
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 8.17 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: 3.5333 seconds (Warm-up)
Chain 3:                2.92538 seconds (Sampling)
Chain 3:                6.45869 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'simple_weight_mod_q8' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000822 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 8.22 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: 3.59817 seconds (Warm-up)
```

```
Chain 4:                3.30563 seconds (Sampling)
Chain 4:                6.9038 seconds (Total)
Chain 4:
```

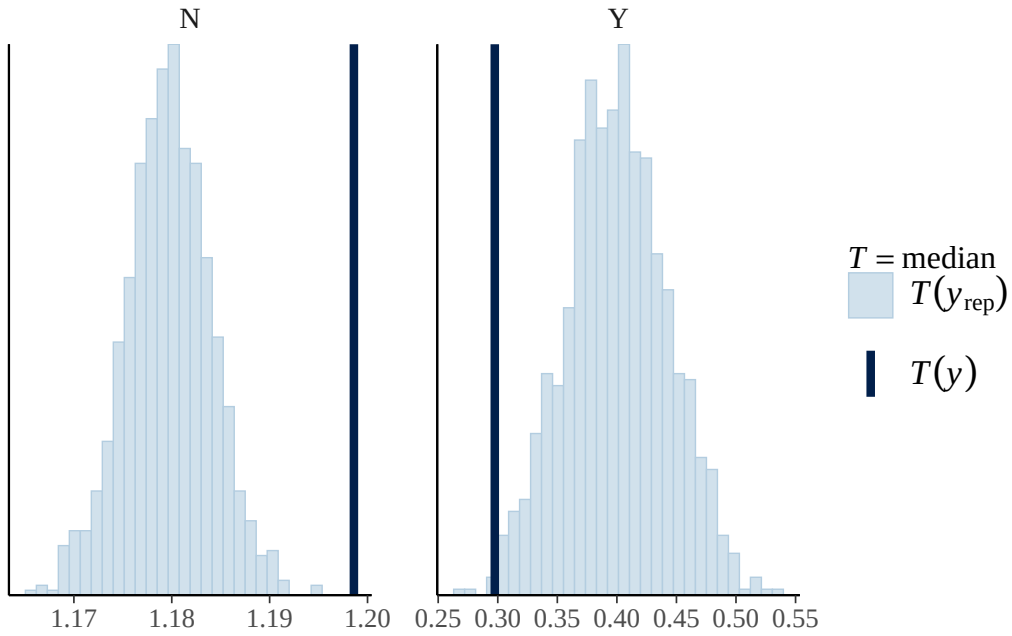
```r
summary(mod3)$summary[c("beta[1]", "beta[2]", "beta[3]", "beta[4]", "beta[5]", "sigma"),]
```

```
              mean      se_mean          sd        2.5%         25%        50%
beta[1] 1.16952830 7.118697e-05 0.002729836 1.163928427 1.167706262 1.16947874
beta[2] 0.56339899 3.574914e-03 0.062187704 0.449412012 0.520151377 0.56145307
beta[3] 0.10248166 1.315911e-04 0.003578246 0.095703010 0.100201349 0.10254455
beta[4] 0.19860142 7.617528e-04 0.012940281 0.174339350 0.189500169 0.19804817
beta[5] 0.01145547 7.571340e-05 0.002563798 0.006399744 0.009790427 0.01138858
sigma   0.16085154 8.227434e-05 0.001847622 0.157442979 0.159578906 0.16081617
               75%      97.5%     n_eff      Rhat
beta[1] 1.17134020 1.17475856 1470.5238 0.9972360
beta[2] 0.60844930 0.69170137  302.6061 1.0061588
beta[3] 0.10486454 0.10937356  739.4139 1.0035159
beta[4] 0.20778748 0.22521140  288.5754 1.0078298
beta[5] 0.01306141 0.01647002 1146.6270 0.9990444
sigma   0.16212224 0.16442880  504.3100 1.0028659
```

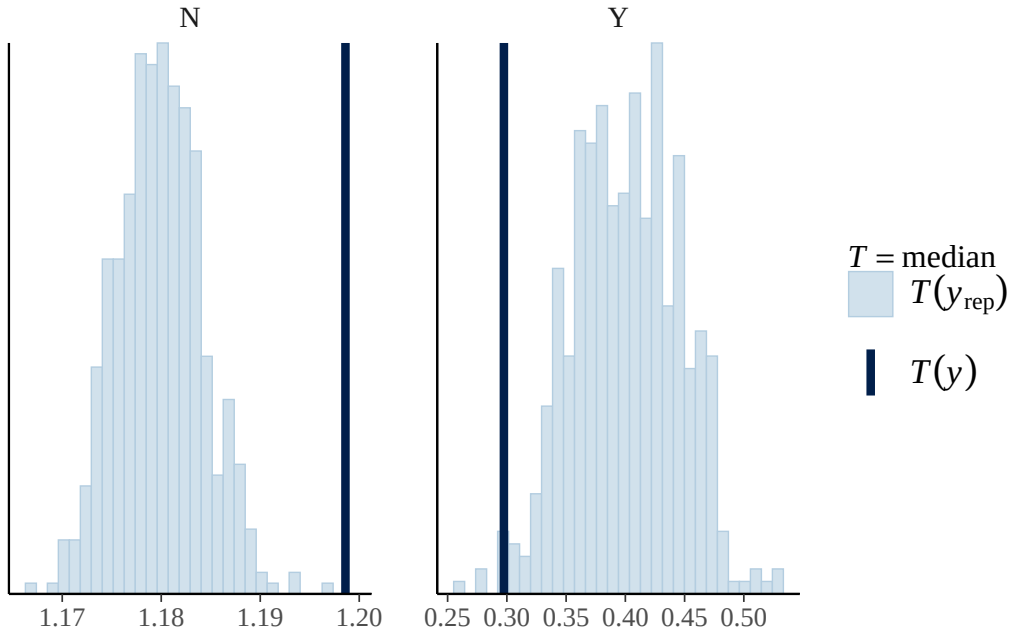
Then I checked the median by prematurity and proportion of births under 2.5kg.

```r
set.seed(1856)
y <- ds$log_weight
yrep3 <- extract(mod3)[["log_weight_rep"]]
```

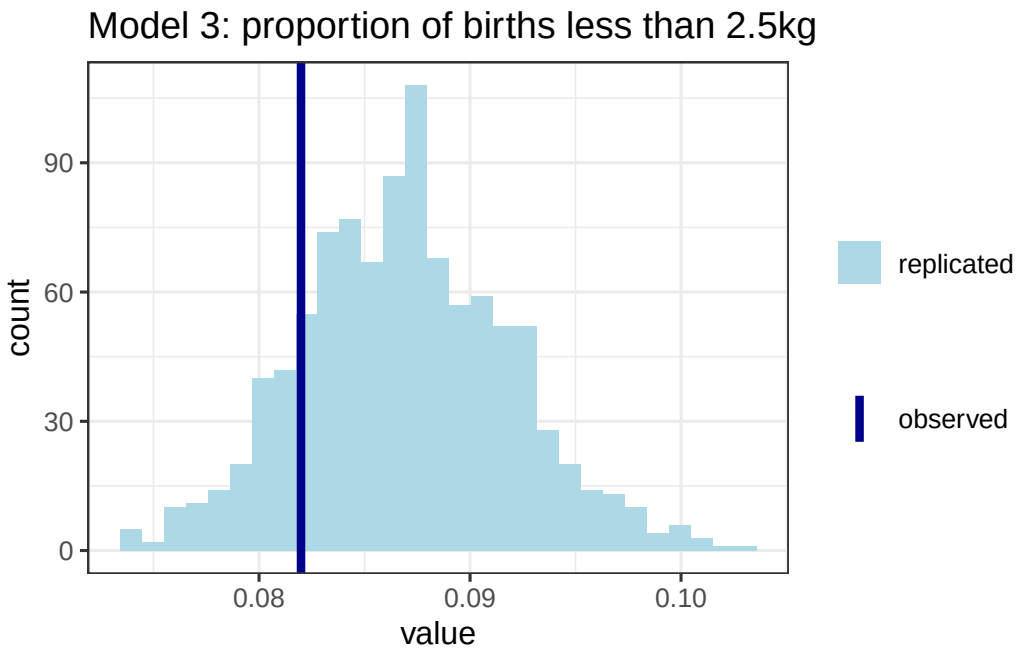
```r
#model3
ppc_stat_grouped(ds$log_weight, yrep3, group = ds$preterm, stat = 'median')
```




```
#model2
ppc_stat_grouped(ds$log_weight, yrep2, group = ds$preterm, stat = 'median')
```

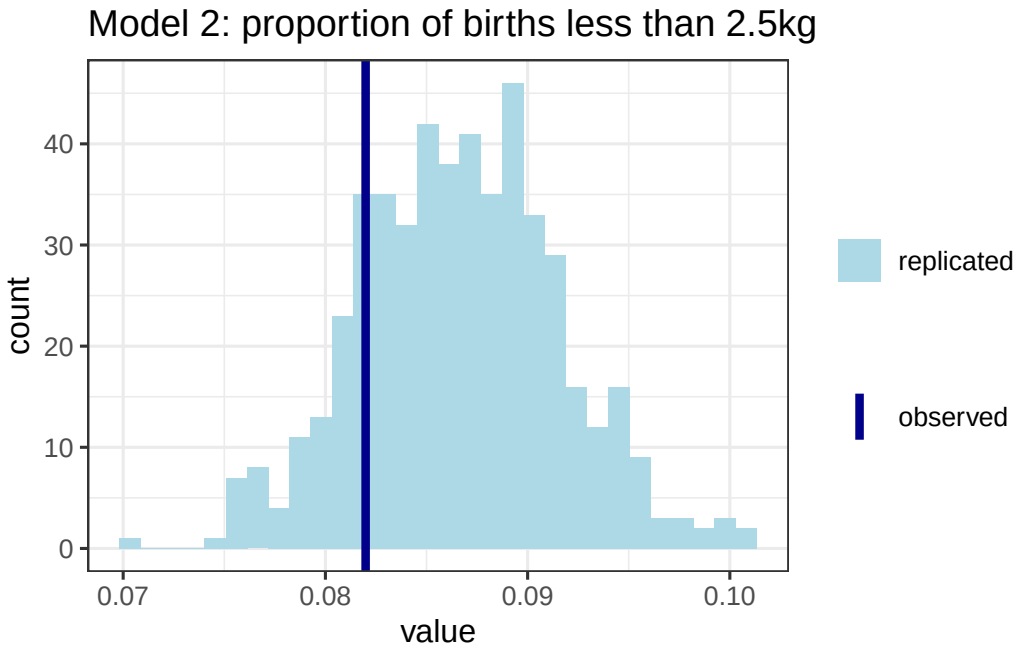


```
# code taken from https://www.monicaalexander.com/posts/2020-28-02-bayes_viz/
t_y <- mean(y<=log(2.5))
t_y_rep_3 <- sapply(1:nrow(yrep3), function(i) mean(yrep3[i,]<=log(2.5)))
t_y_rep_2 <- sapply(1:nrow(yrep2), function(i) mean(yrep2[i,]<=log(2.5)))

ggplot(data = as_tibble(t_y_rep_3), aes(value)) +
    geom_histogram(aes(fill = "replicated")) +
    geom_vline(aes(xintercept = t_y, color = "observed"), lwd = 1.5) +
  ggtitle("Model 3: proportion of births less than 2.5kg") +
  theme_bw(base_size = 12) +
  scale_color_manual(name = "",
                     values = c("observed" = "darkblue"))+
  scale_fill_manual(name = "",
                     values = c("replicated" = "lightblue"))
```




```
ggplot(data = as_tibble(t_y_rep_2), aes(value)) +
    geom_histogram(aes(fill = "replicated")) +
    geom_vline(aes(xintercept = t_y, color = "observed"), lwd = 1.5) +
  ggtitle("Model 2: proportion of births less than 2.5kg") +
  theme_bw(base_size = 12) +
  scale_color_manual(name = "",
```

```
                    values = c("observed" = "darkblue"))+
scale_fill_manual(name = "",
                    values = c("replicated" = "lightblue"))
```



For both median by preterm and proportion of births under 2.5kg, the replicated values are both around the observed. There is no significant difference between model3 (add variable) and model2.