# lab 6: Visualizing the Bayesian Workflow

Anastasia Sokolova

## Introduction

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

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

## The data

Read it in, along with all our packages.

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

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

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

Brief overview of variables:

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

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

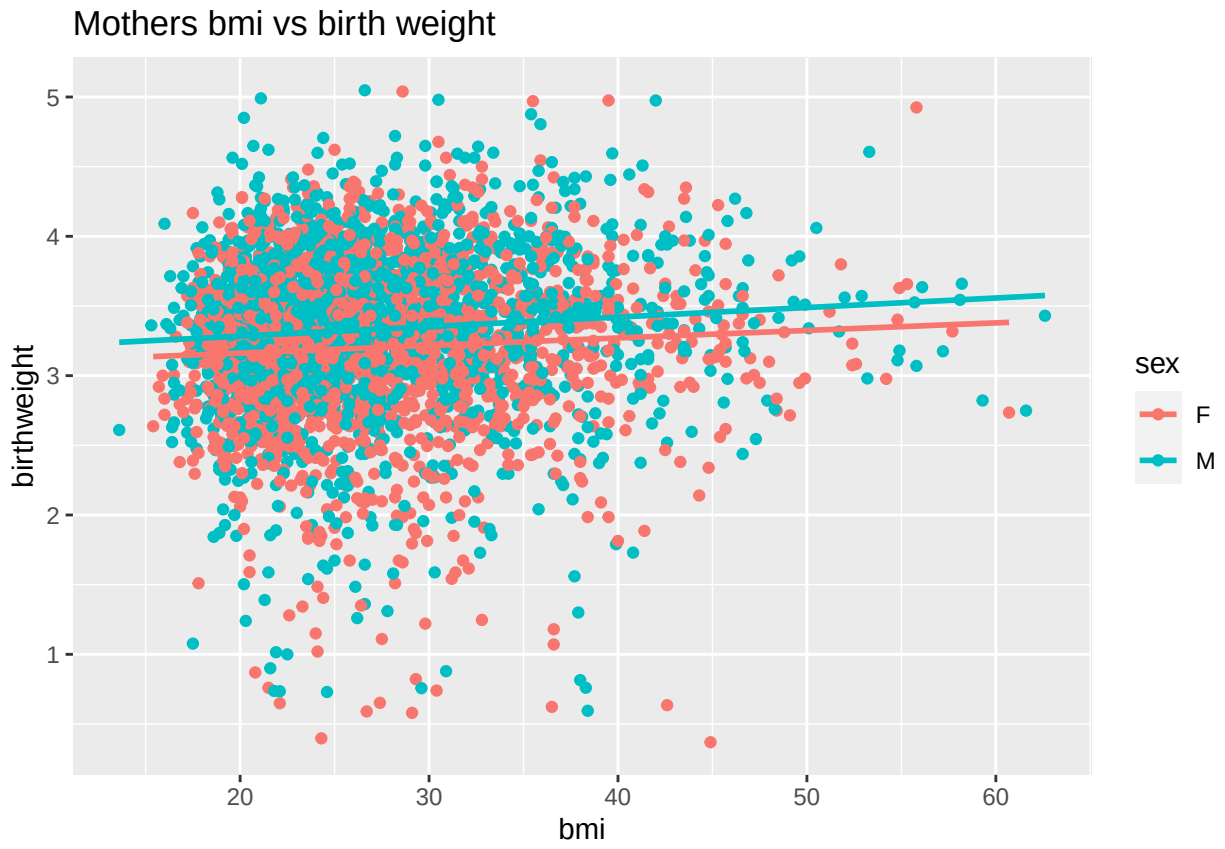
```
ds <- ds %>%
  rename(birthweight = dbwt, gest = combgest) %>%
  mutate(preterm = ifelse(gest<32, "Y", "N")) %>%
  filter(ilive=="Y",gest< 99, birthweight<9.999)
```

```
ds$mracehisp <- as.factor(ds$mracehisp)
head(ds)
```

```
## # A tibble: 6 x 9
##   mager mracehisp meduc   bmi sex    gest birthweight ilive preterm
##   <dbl> <fct>     <dbl> <dbl> <chr> <dbl>       <dbl> <chr> <chr>
## 1    16 2             2  23   M        39        3.18 Y     N
## 2    25 7             2  43.6 M        40        4.14 Y     N
## 3    27 2             3  19.5 F        41        3.18 Y     N
## 4    26 1             3  21.5 F        36        3.40 Y     N
## 5    28 7             2  40.6 F        34        2.71 Y     N
## 6    31 7             3  29.3 M        35        3.52 Y     N
```

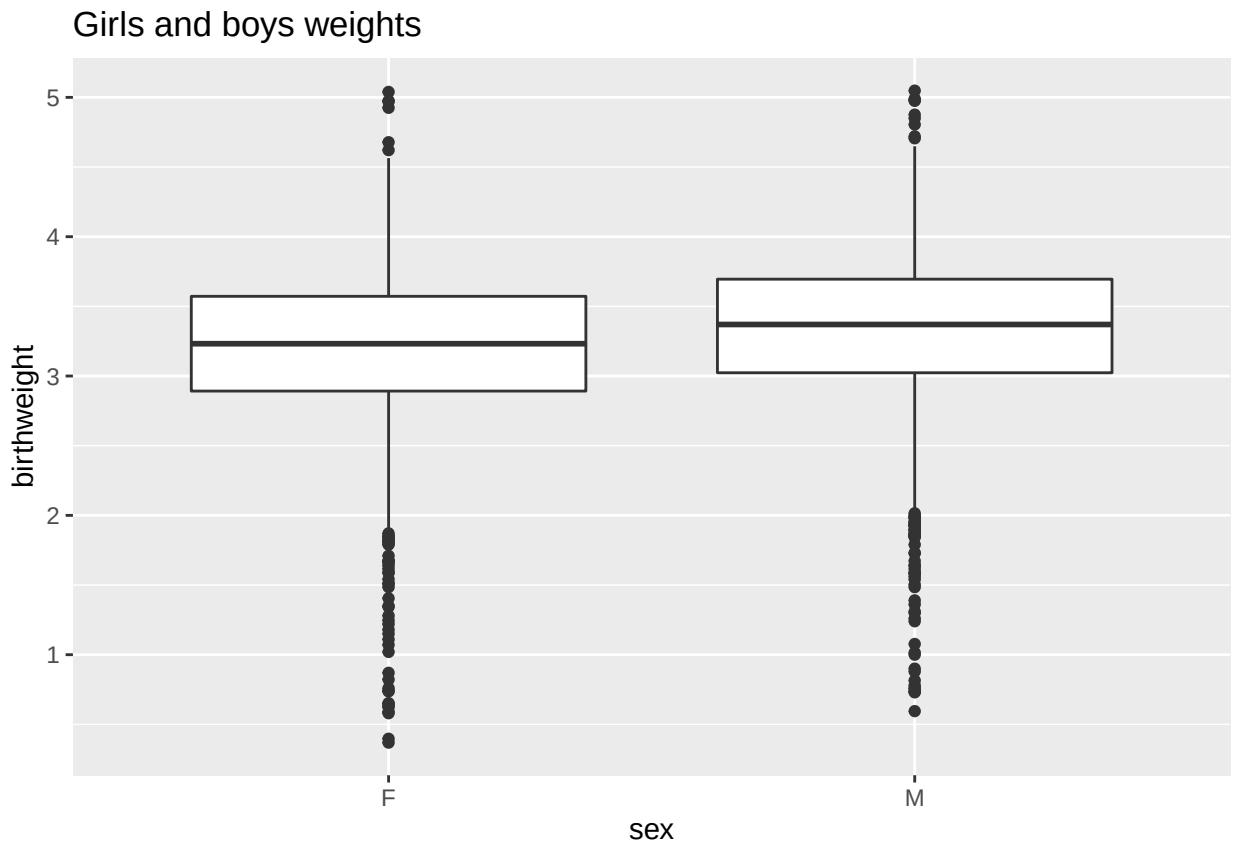
## Question 1

```
ggplot(data = ds%>% filter(bmi < 99.9), aes(x = bmi , y = birthweight, color = sex)) +
  geom_point()+
  geom_smooth(method = 'lm', se = F)+
  ggtitle("Mothers bmi vs birth weight")
```



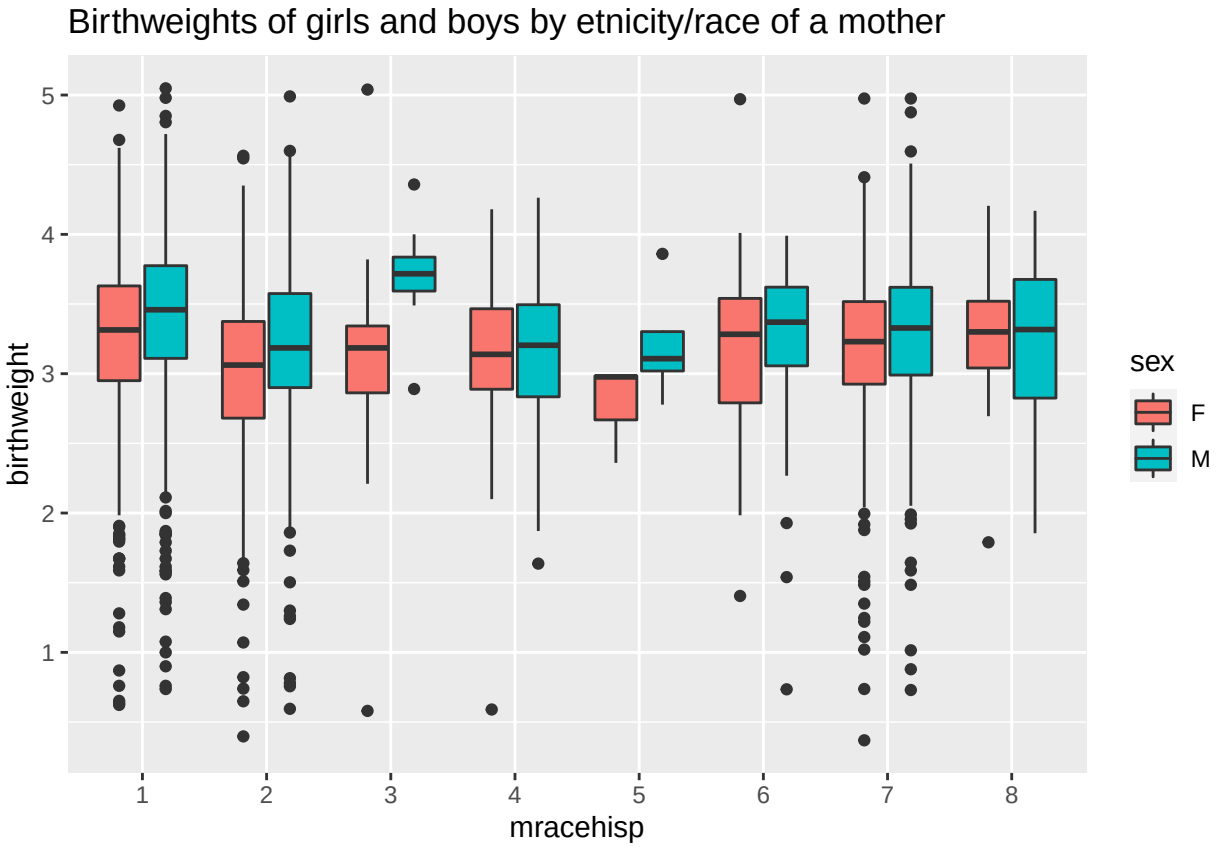
We may observe that there exist slight positive relationship between mother's bmi and baby's weight.

```
ggplot(data = ds, aes(x = sex, y = birthweight)) +
    geom_boxplot()+
  ggtitle('Girls and boys weights')
```




Boy's and girl's weights have similar distributions, with the fact that boy's average is higher.

```
ggplot(data = ds, aes(x = mracehisp, y = birthweight, fill = sex)) +
    geom_boxplot()+
  ggtitle('Birthweights of girls and boys by etnicity/race of a mother')
```




One may observe that distribution of birthweights varies across diffrent race/ethnicity of a mother.

## 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

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

- $y_i$ 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 $\beta$s

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

and for $\sigma$

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

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 $\beta$s and $\sigma$ based on the priors above. Use these values to simulate (log) birth weights from the likelihood specified in Model 1, based on the set of observed gestational weights. Plot the resulting distribution of simulated (log) birth weights. Do 1000 simulations. **Remember the gestational weights should be centered and standardized**.

## 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

```
Sys.setenv(BINPREF = 'C:/rtools40/mingw64/bin/')
```

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

```
##
## SAMPLING FOR MODEL 'simple_weight' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.003 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 30 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: 1.383 seconds (Warm-up)
## Chain 1:                1.017 seconds (Sampling)
## Chain 1:                2.4 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'simple_weight' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.001 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 10 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: 1.089 seconds (Warm-up)
## Chain 2:                1.013 seconds (Sampling)
## Chain 2:                2.102 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'simple_weight' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.001 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 10 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: 1.161 seconds (Warm-up)
## Chain 3:                0.918 seconds (Sampling)
## Chain 3:                2.079 seconds (Total)
```

```
## Chain 3:
##
## SAMPLING FOR MODEL 'simple_weight' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.001 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 10 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: 1.061 seconds (Warm-up)
## Chain 4:                0.984 seconds (Sampling)
## Chain 4:                2.045 seconds (Total)
## Chain 4:
```

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

```
##              mean      se_mean          sd      2.5%       25%       50%
## beta[1] 1.1626507 8.110475e-05 0.002714086 1.1573955 1.1608221 1.1626281
## beta[2] 0.1436061 7.908990e-05 0.002793714 0.1380281 0.1416853 0.1436199
## sigma   0.1688517 1.090725e-04 0.002019961 0.1650908 0.1674982 0.1687796
##               75%     97.5%     n_eff      Rhat
## beta[1] 1.1645352 1.1678892 1119.8364 0.9973582
## beta[2] 0.1454956 0.1489020 1247.7331 0.9981180
## sigma   0.1700702 0.1729636  342.9693 1.0067923
```

## Question 3

```
ds$preterm <- 1*(ds$preterm == 'Y')
ds$log_gest_c_preterm <- ds$log_gest_c*ds$preterm

stan_data_2 <- list(N = nrow(ds),
                    log_weight = ds$log_weight,
                    log_gest = ds$log_gest_c,
                    z = ds$preterm,
                    interac = ds$log_gest_c_preterm)
```

```
mod2_my <- stan(data = stan_data_2,
             file = "model_2_weight.stan",
             iter = 500,
             seed = 243)
```

```
##
```

```
## SAMPLING FOR MODEL 'model_2_weight' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.005 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 50 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: 7.277 seconds (Warm-up)
## Chain 1:                5.075 seconds (Sampling)
## Chain 1:                12.352 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'model_2_weight' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.002 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 20 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: 6.86 seconds (Warm-up)
## Chain 2:                5.742 seconds (Sampling)
## Chain 2:                12.602 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'model_2_weight' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.001 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 10 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: 5.54 seconds (Warm-up)
## Chain 3:                3.697 seconds (Sampling)
## Chain 3:                9.237 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'model_2_weight' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.001 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 10 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: 5.327 seconds (Warm-up)
## Chain 4:                4.257 seconds (Sampling)
## Chain 4:                9.584 seconds (Total)
## Chain 4:
```

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

```
summary(mod2_my)$summary[c("beta[1]", "beta[2]", "beta[3]", "beta[4]", "sigma"),]
```

```
##              mean      se_mean          sd       2.5%        25%       50%
## beta[1] 1.1696110 7.245593e-05 0.002565397 1.16454954 1.16795255 1.1694902
## beta[2] 0.1019057 1.122666e-04 0.003470002 0.09529398 0.09952048 0.1018066
## beta[3] 0.5570611 3.735631e-03 0.067108729 0.43564438 0.50976342 0.5588179
## beta[4] 0.1969881 7.889378e-04 0.013599443 0.17170899 0.18773902 0.1970246
```

```
## sigma   0.1613562 8.240442e-05 0.001923409 0.15776716 0.16000880 0.1613946
##                75%     97.5%     n_eff      Rhat
## beta[1] 1.1713951 1.1744215 1253.6064 0.9999657
## beta[2] 0.1042732 0.1087897  955.3413 0.9981139
## beta[3] 0.6036398 0.6898052  322.7231 0.9980738
## beta[4] 0.2056903 0.2240259  297.1370 0.9983563
## sigma   0.1627059 0.1651897  544.8067 1.0042140
```

### Question 4

For reference I have uploaded some model 2 results. Check your results are similar. ($\beta_2$ relates to gestational age, $\beta_3$ relates to preterm, $\beta_4$ is the interaction).

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

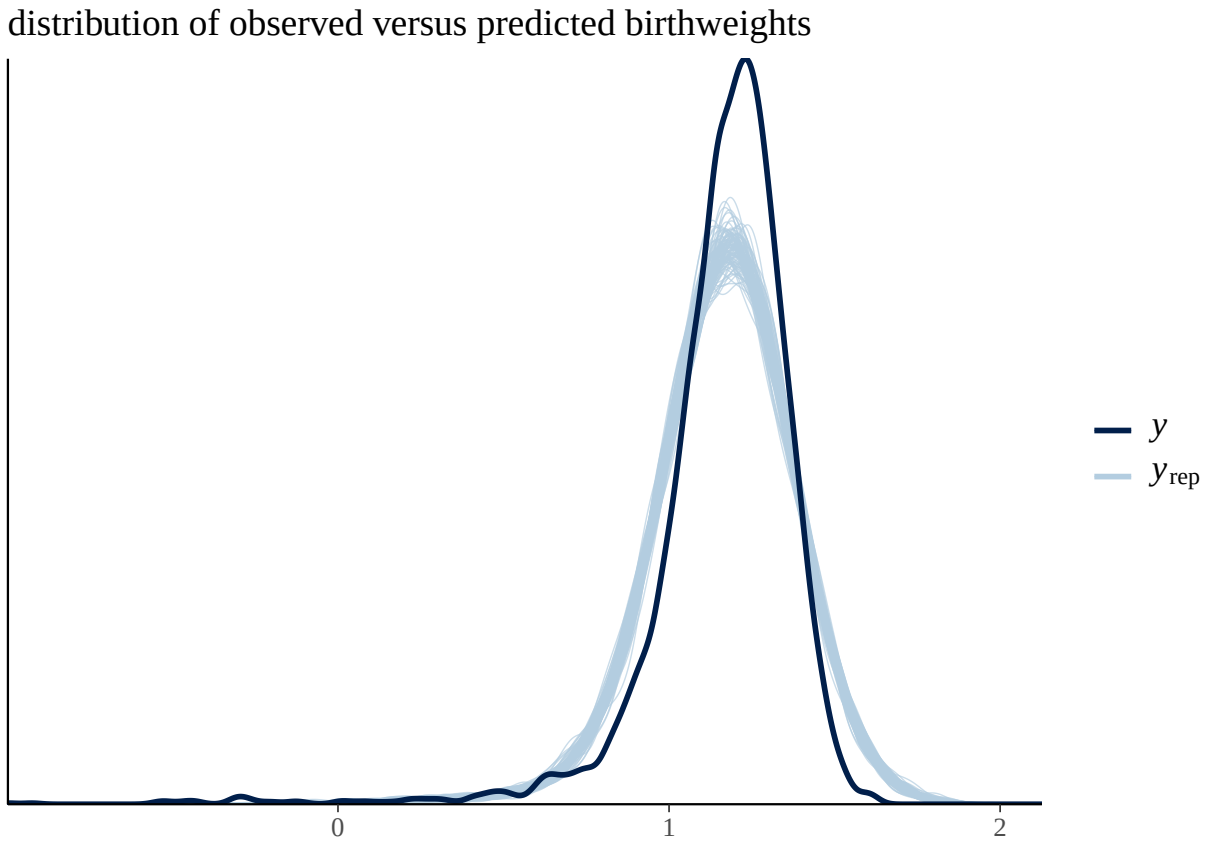
```
##              mean      se_mean          sd       2.5%        25%       50%
## beta[1] 1.1697241 1.385590e-04 0.002742186 1.16453578 1.16767109 1.1699278
## beta[2] 0.5563133 5.835253e-03 0.058054991 0.43745504 0.51708255 0.5561553
## beta[3] 0.1020960 1.481816e-04 0.003669476 0.09459462 0.09997153 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
```

```
mod2 <- mod2_my
```

# PPCs

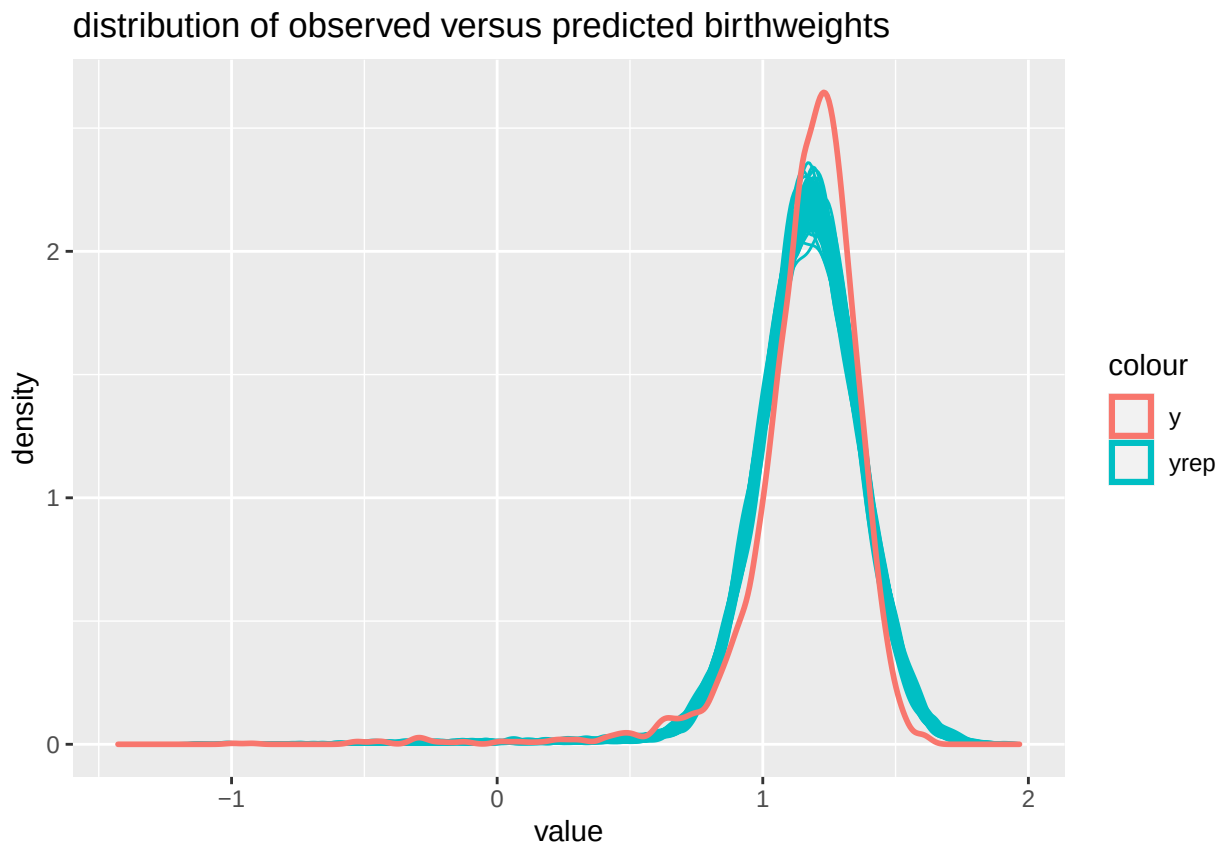
Now we've run two candidate models let's do some posterior predictive checks. The `bayesplot` package has a lot of inbuilt graphing functions to do this. For example, let's plot the distribution of our data (y) against 100 different datasets drawn from the posterior predictive distribution:

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




### Question 5

```
d <- ppc_data(y, yrep2[samp100, ])
ggplot() +
geom_density(
  data = d %>% filter(!is_y),
  aes(x = value, group=rep_id, color = "yrep"),
  geom = "line",
  position = "identity",
) +
geom_density(
  data = d %>% filter(is_y),
  aes(x = value, color = "y"),
  geom = "line",
  position = "identity",
  lineend = "round",
  size = 1,
) +
ggtitle("distribution of observed versus predicted birthweights")
```




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`)

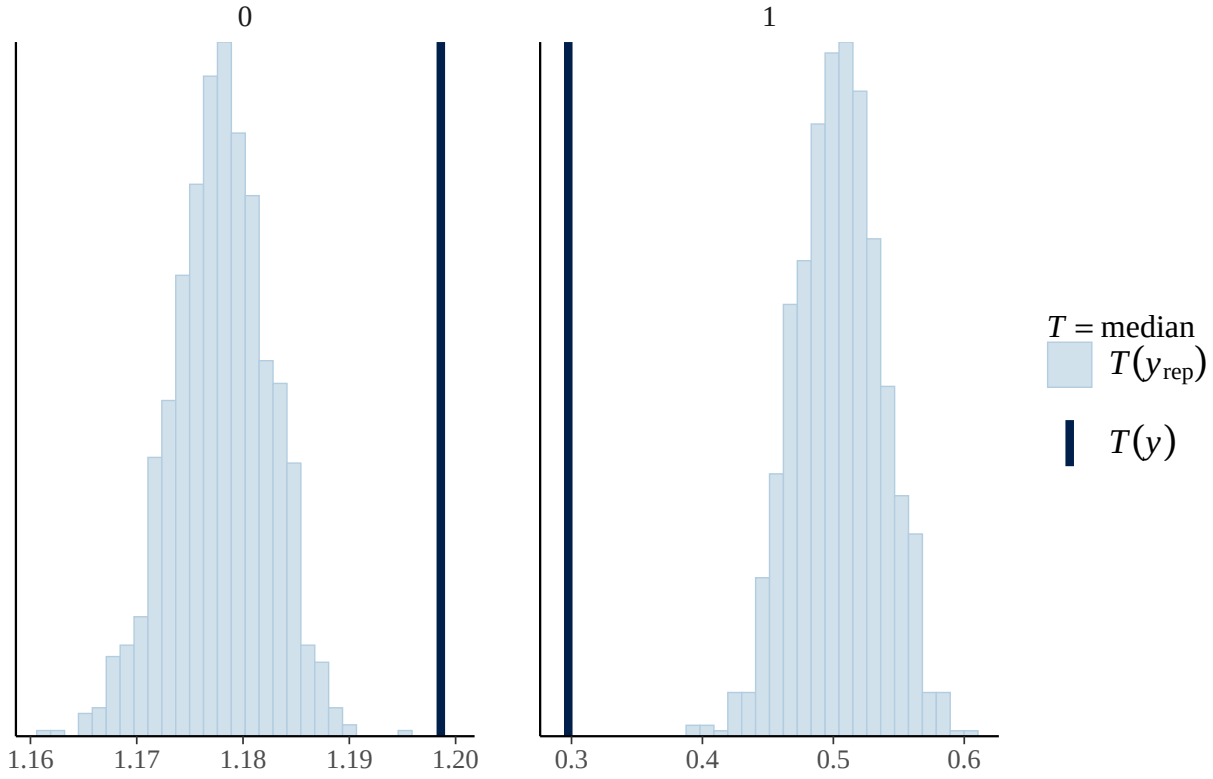
### Test statistics

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')+
  ggtitle('Model 1')
```

Model 1




## Question 6

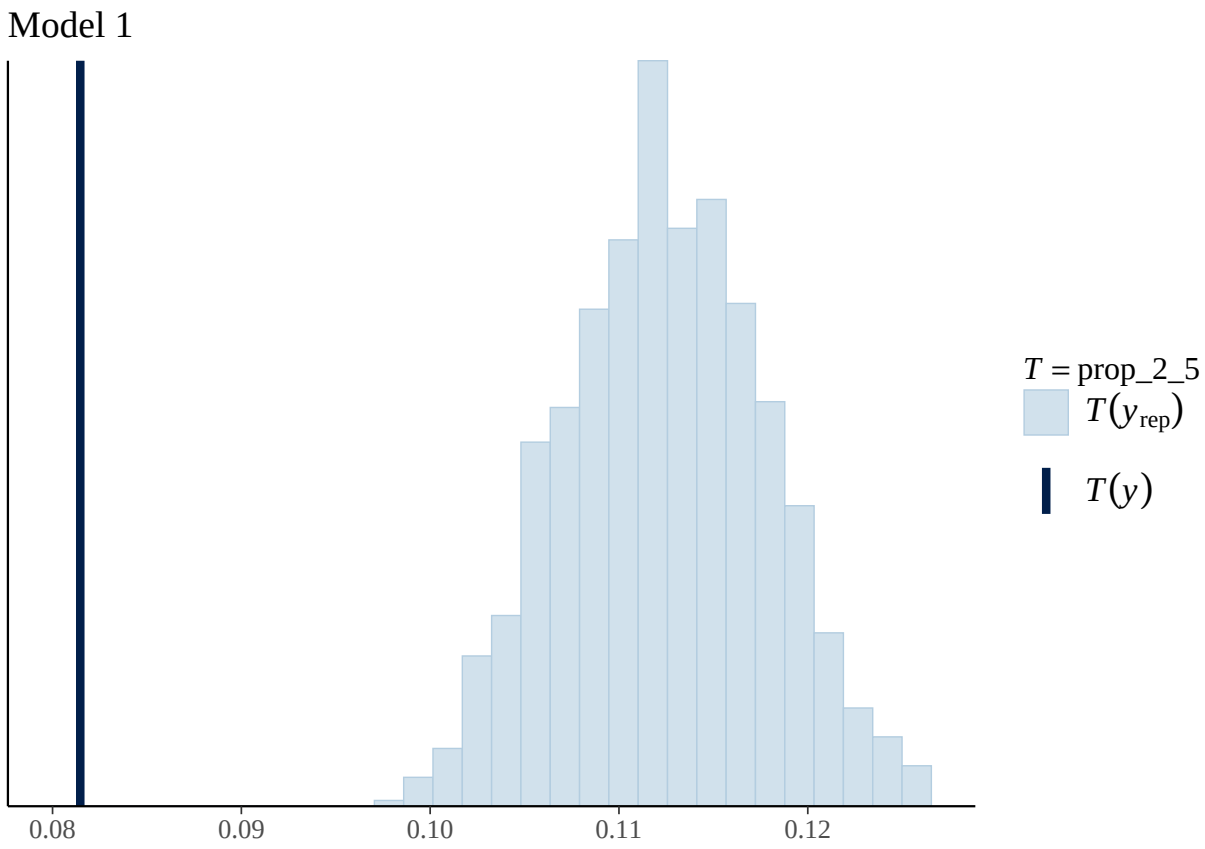
Use a test statistic of the proportion of births under 2.5kg.

```
prop_2_5 <- function(x) mean(x < log(2.5))
prop_2_5(y)
```

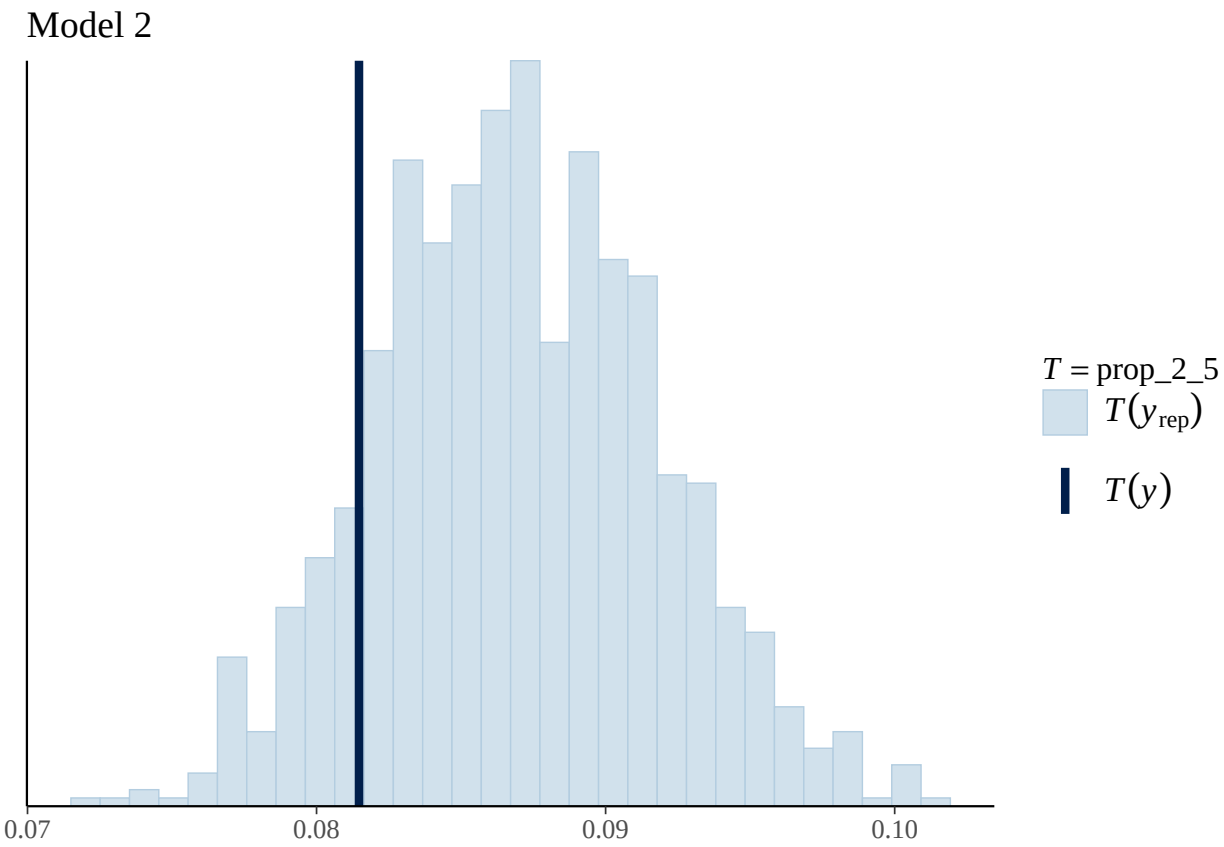
```
## [1] 0.08146799
```

Calculate the test statistic for the data, and the posterior predictive samples for both models, and plot the comparison (one plot per model).

```
ppc_stat(ds$log_weight, yrep1, stat = "prop_2_5")+
  ggtitle('Model 1')
```




```
ppc_stat(ds$log_weight, yrep2, stat = "prop_2_5")+
  ggtitle('Model 2')
```




# 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   1377.1  72.6
## p_loo         9.8   1.5
## looic     -2754.2 145.3
## ------
## 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   1552.5  69.7
## p_loo        15.5   2.4
## looic     -3105.1 139.4
## ------
## 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.
```

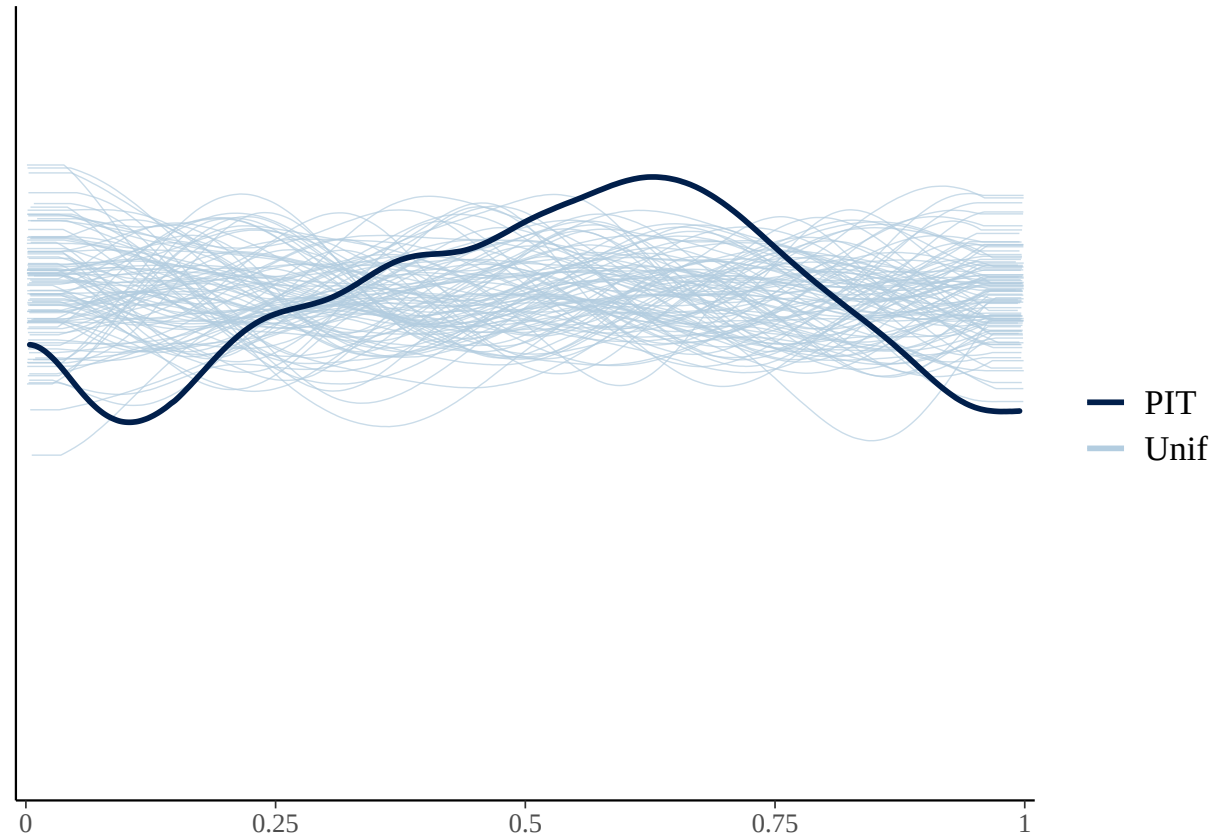
Comparing the two models tells us Model 2 is better:

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

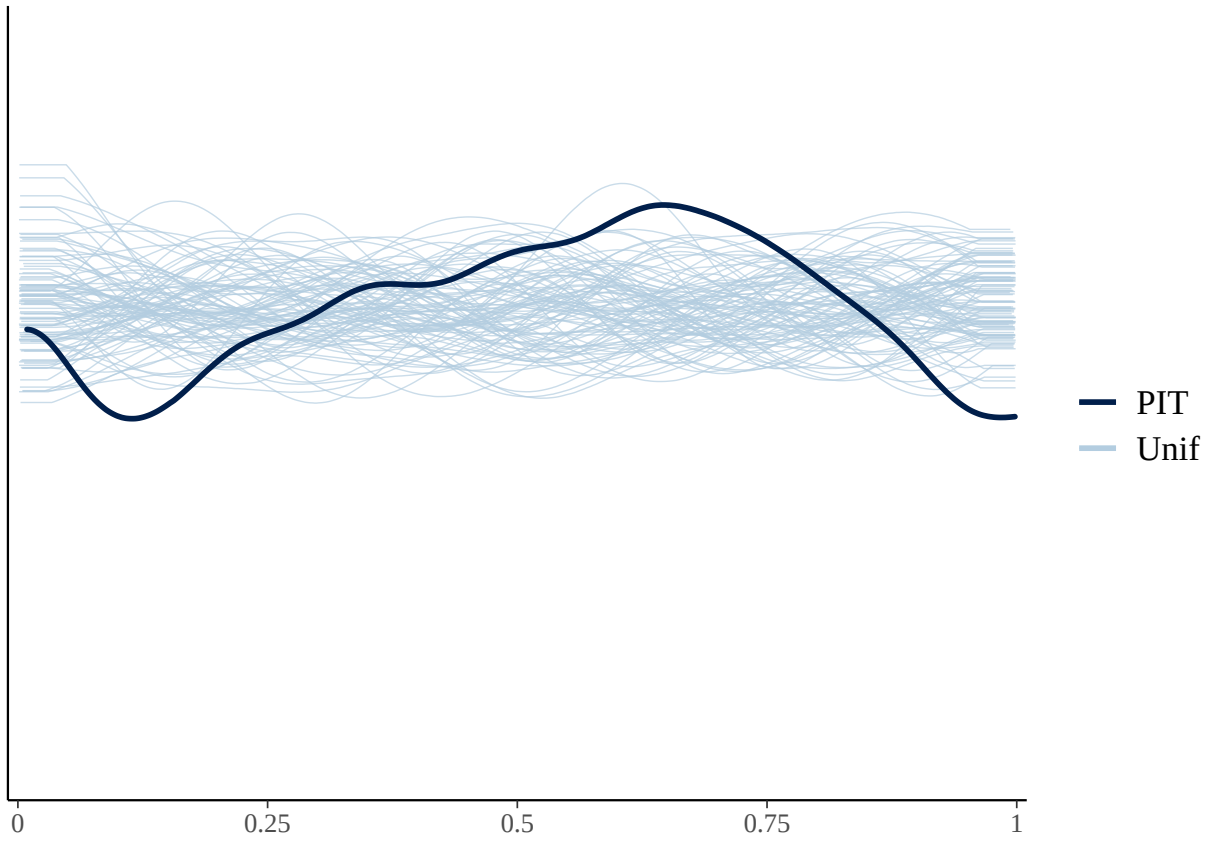
```
##        elpd_diff se_diff
## model2    0.0       0.0
## model1 -175.5      36.3
```

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

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