

Week 6: Visualizing the Bayesian Workflow

24/02/23

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/nativity/2017/natl2017.pdf> page 15
- `meduc` mum's education see here for codes: <https://data.nber.org/nativity/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

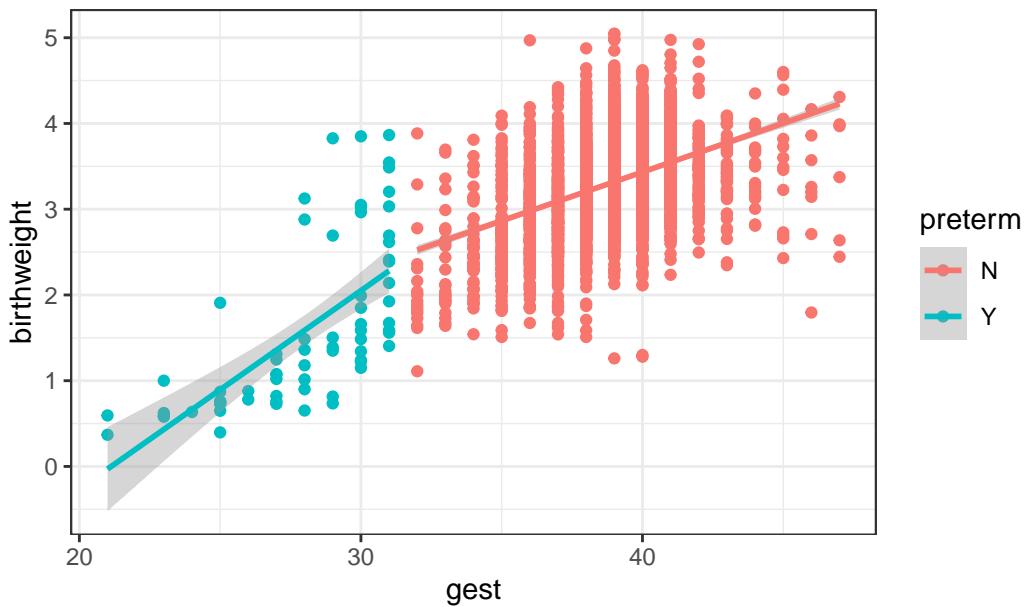
Use plots or tables to show three interesting observations about the data. Remember:

- 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 = gest, y = birthweight, color = preterm, group = preterm) +
  geom_point() +
  theme_bw() +
  geom_smooth(method = "lm") +
  labs(title = "Weight vs Gestational Age")
```

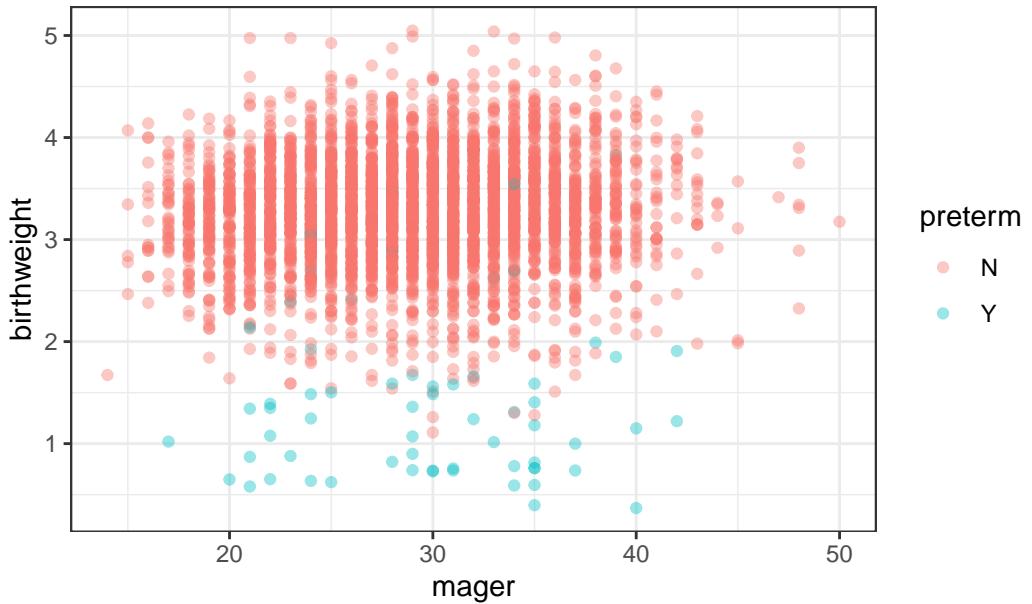
Weight vs Gestational Age



I replicated the scatter plots in the lecture. The plot shows the gestational age in weeks and baby's birth weight. We see that, there is a increasing trend as gestational age in weeks increase. However, the slope is different whether or not the baby is preterm.

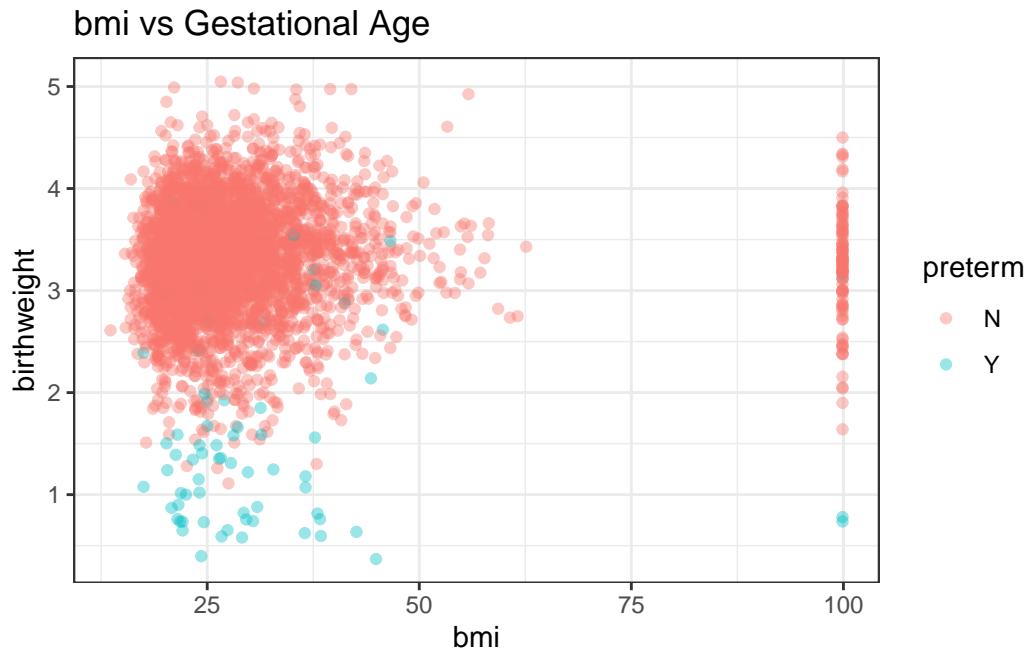
```
ds %>% ggplot() + aes(x = mager, y = birthweight, color = preterm) +
  geom_point(alpha = 0.4) +
  theme_bw() +
  labs(title = "Weight vs Gestational Age")
```

Weight vs Gestational Age



This plot shows the mother's age on the x-axis and baby's birth weight on the y axis. Based on the plot, it is not obvious whether or not the mother's age will affect the birth weight, but it is more clear that we have unbalanced preterm data.

```
ds %>% ggplot() + aes(x = bmi, y = birthweight, color = preterm) +
  geom_point(alpha = 0.4) +
  theme_bw() +
  labs(title = "bmi vs Gestational Age")
```



There is some problem with the BMI coding, there are many observation with BMI of 100.

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 β s

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

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

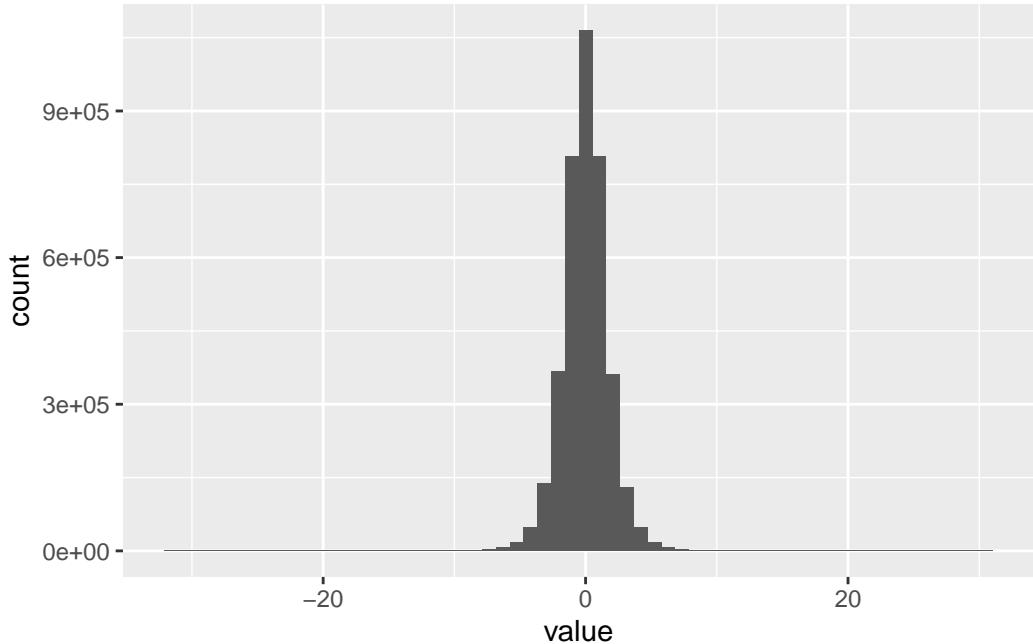
For Model 1, simulate values of β s and σ based on the priors above. Do 1000 simulations. Use these values to simulate (log) birth weights from the likelihood specified in Model 1, based on the set of observed gestational weights. **Remember the gestational weights should be centered and standardized.**

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

```
set.seed(1)
beta1 = rnorm(1000)
beta2 = rnorm(1000)
sigma = abs(rnorm(1000))
gestational <- ds %>% mutate(log_z_gest = scale(log(gest))) %>% select(log_z_gest)

for (i in 1:1000){
  mu <- beta1[i] + beta2[i] * gestational$log_z_gest
  gestational[i+1] = rnorm(3842, mean = mu, sd = sigma[i])
}

gestational %>% select(-log_z_gest) %>% pivot_longer(cols = everything()) %>%
  ggplot(aes(x= value)) +
  geom_histogram(bins = 60)
```



```

sim10 <- gestational[1:11] %>%
  pivot_longer(cols = starts_with("..."),
               values_to = "Sim_num",
               values_to = "log_birthweights")
sim10

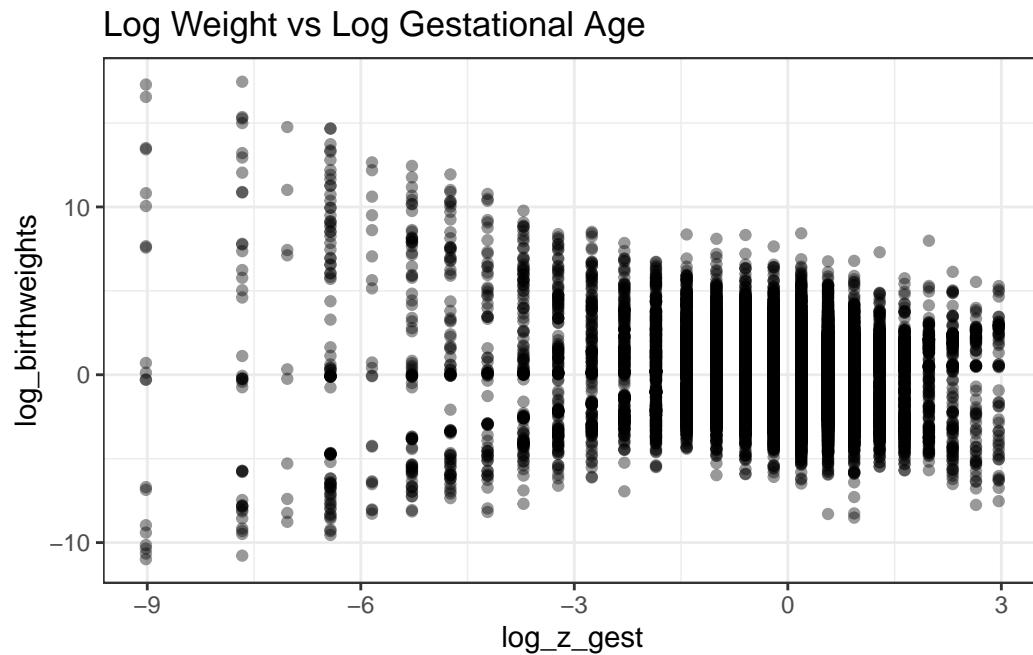
```

```

# A tibble: 38,420 x 3
  log_z_gest[,1] Sim_num log_birthweights
  <dbl> <chr>          <dbl>
1 0.188 ...2      0.242
2 0.188 ...3     -0.916
3 0.188 ...4     -5.15 
4 0.188 ...5      2.23 
5 0.188 ...6      0.305
6 0.188 ...7      0.136
7 0.188 ...8      0.649
8 0.188 ...9      1.07 
9 0.188 ...10     -0.0466
10 0.188 ...11     0.306
# ... with 38,410 more rows

```

```
sim10 %>% ggplot() + aes(x = log_z_gest, y = log_birthweights) +
  geom_point(alpha = 0.4) +
  theme_bw() +
  labs(title = "Log Weight vs Log 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.

```
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)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000296 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.96 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: 0.399 seconds (Warm-up)
Chain 1:                 0.32 seconds (Sampling)
Chain 1:                 0.719 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000151 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.51 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: 0.385 seconds (Warm-up)
Chain 2: 0.354 seconds (Sampling)
Chain 2: 0.739 seconds (Total)
Chain 2:
```

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).

```
Chain 3:
Chain 3: Gradient evaluation took 0.000139 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.39 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: 0.378 seconds (Warm-up)
Chain 3: 0.322 seconds (Sampling)
Chain 3: 0.7 seconds (Total)
Chain 3:
```

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).

```
Chain 4:
Chain 4: Gradient evaluation took 0.00018 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.8 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: 0.362 seconds (Warm-up)
Chain 4:           0.325 seconds (Sampling)
Chain 4:           0.687 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		

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((log(37) - mean(log(ds$gest)))/sd(log(ds$gest))*0.1437529 + 1.1624783)
```

```
[1] 2.935874
```

Using the estimated posterior mean, the expected birth weight of a baby who was born at a gestational age of 37 weeks is 2.935kg

Question 4

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

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

stan_data <- list(N = nrow(ds),
                  log_weight = ds$log_weight,
                  log_gest = ds$log_gest_c,
                  preterm = ifelse(ds$preterm == "Y", 1, 0),
                  interaction = ds$log_gest_c*preterm)

mod2 <- stan(data = stan_data,
              file = here("code/models/simple_weight_q4.stan"),
              iter = 1000,
              seed = 243)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000748 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 7.48 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration: 1 / 1000 [  0%] (Warmup)
Chain 1: Iteration: 100 / 1000 [ 10%] (Warmup)
Chain 1: Iteration: 200 / 1000 [ 20%] (Warmup)
Chain 1: Iteration: 300 / 1000 [ 30%] (Warmup)
Chain 1: Iteration: 400 / 1000 [ 40%] (Warmup)
Chain 1: Iteration: 500 / 1000 [ 50%] (Warmup)
Chain 1: Iteration: 501 / 1000 [ 50%] (Sampling)
Chain 1: Iteration: 600 / 1000 [ 60%] (Sampling)
Chain 1: Iteration: 700 / 1000 [ 70%] (Sampling)
Chain 1: Iteration: 800 / 1000 [ 80%] (Sampling)
Chain 1: Iteration: 900 / 1000 [ 90%] (Sampling)
Chain 1: Iteration: 1000 / 1000 [100%] (Sampling)
Chain 1:
```

```

Chain 1: Elapsed Time: 3.987 seconds (Warm-up)
Chain 1:           2.872 seconds (Sampling)
Chain 1:           6.859 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000607 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 6.07 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration: 1 / 1000 [ 0%] (Warmup)
Chain 2: Iteration: 100 / 1000 [ 10%] (Warmup)
Chain 2: Iteration: 200 / 1000 [ 20%] (Warmup)
Chain 2: Iteration: 300 / 1000 [ 30%] (Warmup)
Chain 2: Iteration: 400 / 1000 [ 40%] (Warmup)
Chain 2: Iteration: 500 / 1000 [ 50%] (Warmup)
Chain 2: Iteration: 501 / 1000 [ 50%] (Sampling)
Chain 2: Iteration: 600 / 1000 [ 60%] (Sampling)
Chain 2: Iteration: 700 / 1000 [ 70%] (Sampling)
Chain 2: Iteration: 800 / 1000 [ 80%] (Sampling)
Chain 2: Iteration: 900 / 1000 [ 90%] (Sampling)
Chain 2: Iteration: 1000 / 1000 [100%] (Sampling)
Chain 2:
Chain 2: Elapsed Time: 4.006 seconds (Warm-up)
Chain 2:           4.056 seconds (Sampling)
Chain 2:           8.062 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000471 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 4.71 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration: 1 / 1000 [ 0%] (Warmup)
Chain 3: Iteration: 100 / 1000 [ 10%] (Warmup)
Chain 3: Iteration: 200 / 1000 [ 20%] (Warmup)
Chain 3: Iteration: 300 / 1000 [ 30%] (Warmup)
Chain 3: Iteration: 400 / 1000 [ 40%] (Warmup)
Chain 3: Iteration: 500 / 1000 [ 50%] (Warmup)

```

```

Chain 3: Iteration: 501 / 1000 [ 50%] (Sampling)
Chain 3: Iteration: 600 / 1000 [ 60%] (Sampling)
Chain 3: Iteration: 700 / 1000 [ 70%] (Sampling)
Chain 3: Iteration: 800 / 1000 [ 80%] (Sampling)
Chain 3: Iteration: 900 / 1000 [ 90%] (Sampling)
Chain 3: Iteration: 1000 / 1000 [100%] (Sampling)
Chain 3:
Chain 3:   Elapsed Time: 4.073 seconds (Warm-up)
Chain 3:           3.988 seconds (Sampling)
Chain 3:           8.061 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000514 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 5.14 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration: 1 / 1000 [ 0%] (Warmup)
Chain 4: Iteration: 100 / 1000 [ 10%] (Warmup)
Chain 4: Iteration: 200 / 1000 [ 20%] (Warmup)
Chain 4: Iteration: 300 / 1000 [ 30%] (Warmup)
Chain 4: Iteration: 400 / 1000 [ 40%] (Warmup)
Chain 4: Iteration: 500 / 1000 [ 50%] (Warmup)
Chain 4: Iteration: 501 / 1000 [ 50%] (Sampling)
Chain 4: Iteration: 600 / 1000 [ 60%] (Sampling)
Chain 4: Iteration: 700 / 1000 [ 70%] (Sampling)
Chain 4: Iteration: 800 / 1000 [ 80%] (Sampling)
Chain 4: Iteration: 900 / 1000 [ 90%] (Sampling)
Chain 4: Iteration: 1000 / 1000 [100%] (Sampling)
Chain 4:
Chain 4:   Elapsed Time: 4.004 seconds (Warm-up)
Chain 4:           4.264 seconds (Sampling)
Chain 4:           8.268 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]	1.1695606	5.259778e-05	0.002619877	1.16440944	1.16793892	1.1696211

```

beta[2] 0.1019187 7.460161e-05 0.003372707 0.09534751 0.09970971 0.1019299
beta[3] 0.5617196 3.024670e-03 0.063523319 0.43251124 0.51932707 0.5633255
beta[4] 0.1980411 6.329651e-04 0.013271075 0.17020939 0.18952383 0.1979300
sigma   0.1612523 5.496825e-05 0.001828663 0.15774082 0.16005586 0.1612095
    75%     97.5%      n_eff      Rhat
beta[1] 1.1712320 1.1748836 2481.0005 0.9994273
beta[2] 0.1041208 0.1085593 2043.9053 0.9994972
beta[3] 0.6054852 0.6841378 441.0729 1.0125540
beta[4] 0.2071063 0.2238445 439.5949 1.0107559
sigma   0.1624835 0.1648295 1106.7345 1.0016836

```

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		

The result is similar.

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:

```

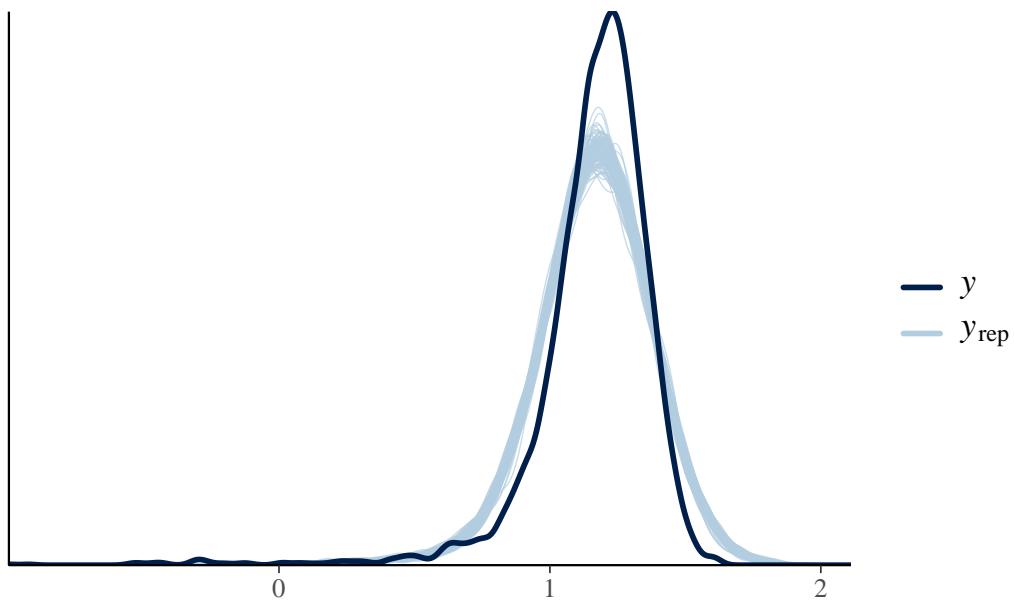
set.seed(1856)
y <- ds$log_weight
yrep1 <- extract(mod1)[["log_weight_rep"]]
yrep2 <- extract(mod2)[["log_weight_rep"]]
dim(yrep1)

[1] 1000 3842

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

```

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`)

```

samp100_2 <- sample(nrow(yrep2), 100)
samples <- as_tibble(t(yrep2[samp100_2, ]))
log_weight <- as_tibble(ds$log_weight)

```

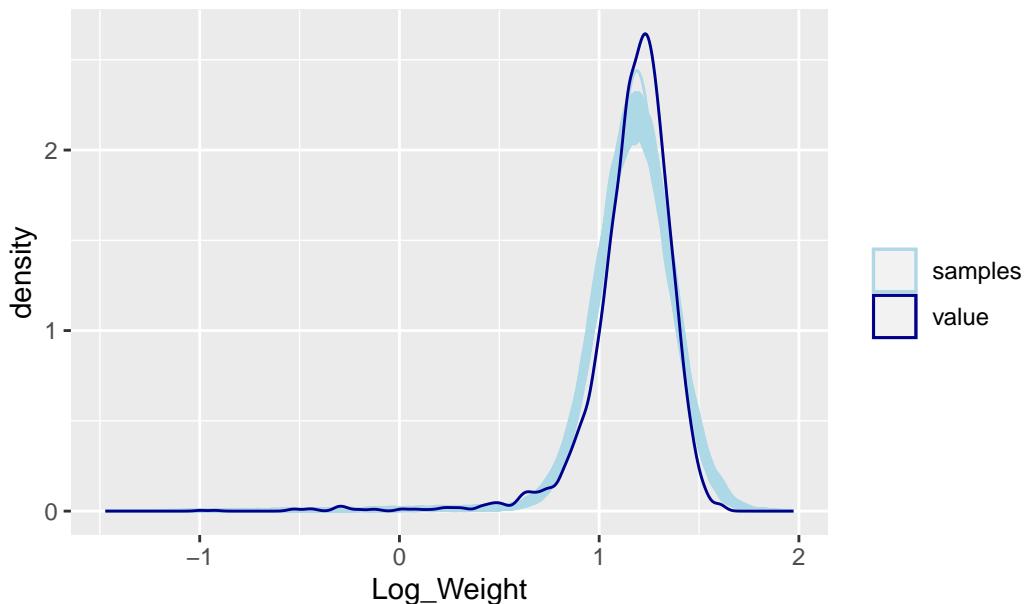
```

samples <- cbind(log_weight, samples)

samples %>%
  pivot_longer( everything(), names_to = "simulate", values_to = "Log_Weight" ) %>%
  ggplot(aes(Log_Weight, group = simulate)) +
  geom_density(alpha = 0.2, aes(color = "samples")) +
  geom_density(data = ds %>%
                mutate(simulate = "value"), aes(x = log(birthweight), col = "value")) +
  scale_color_manual(name = "", 
                     values = c("value" = "darkblue", "samples" = "lightblue")) +
  ggtitle("Distribution of observed vs predicted birthweights")

```

Distribution of observed vs predicted birthweights

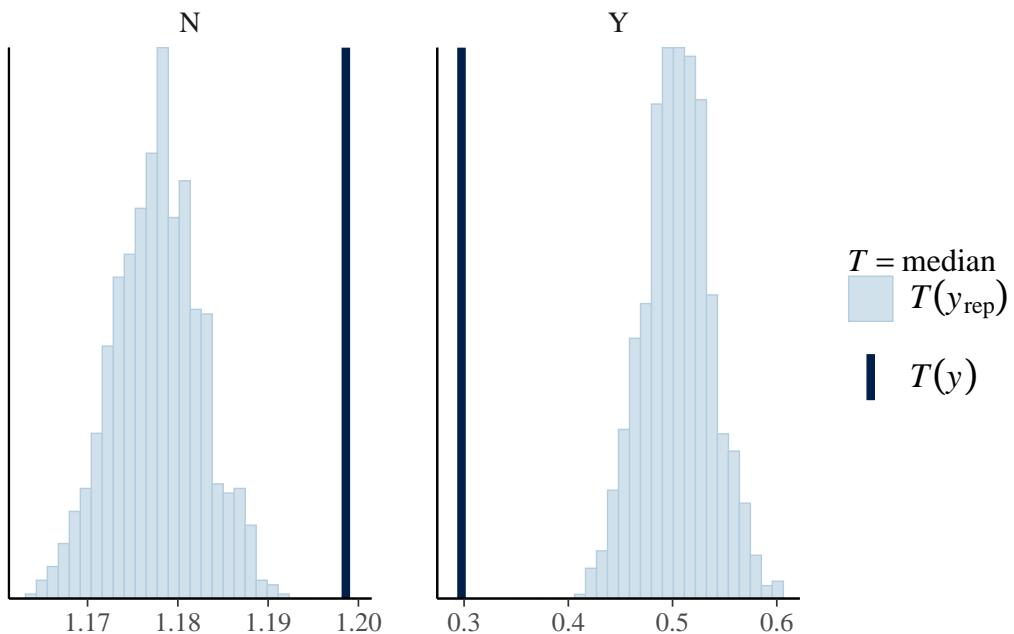


Test statistics

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

E.g. medians by prematurity for Model 1

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



Question 7

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

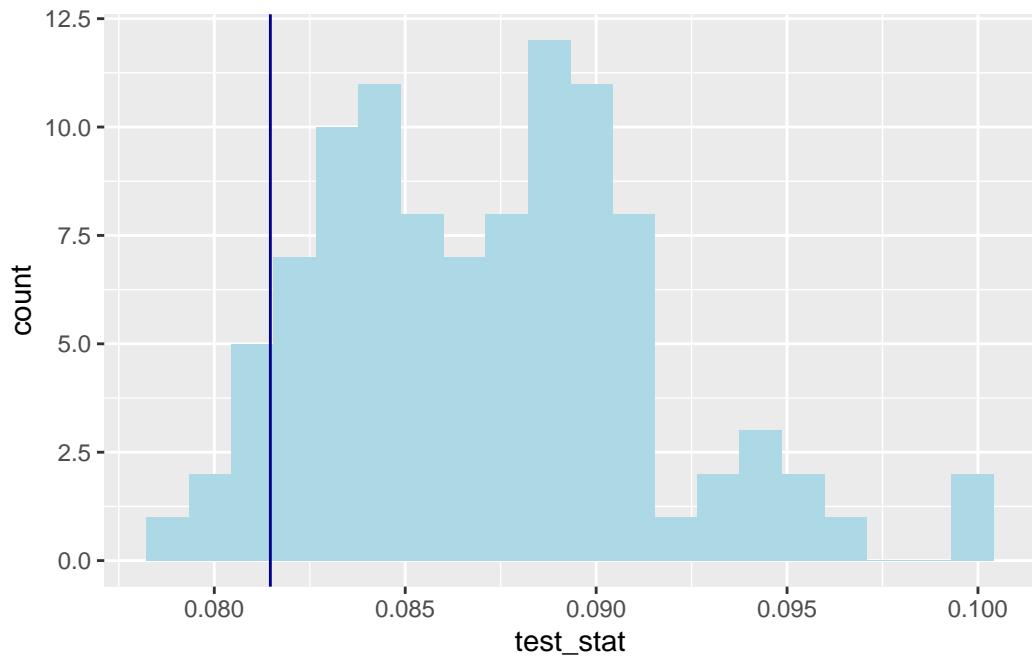
```
mean(ds$birthweight < 2.5)
```

```
[1] 0.08146799
```

Model 2:

```
samples %>%
  pivot_longer( everything(), names_to = "simulate", values_to = "Log_Weight" ) %>%
  group_by(simulate) %>%
  summarize(test_stat = mean(exp(Log_Weight) < 2.5)) %>%
  ggplot(aes(x = test_stat)) +
  geom_histogram(bins = 20, fill = "lightblue") +
```

```
geom_vline(xintercept = mean(ds$birthweight < 2.5), color = "darkblue")
```



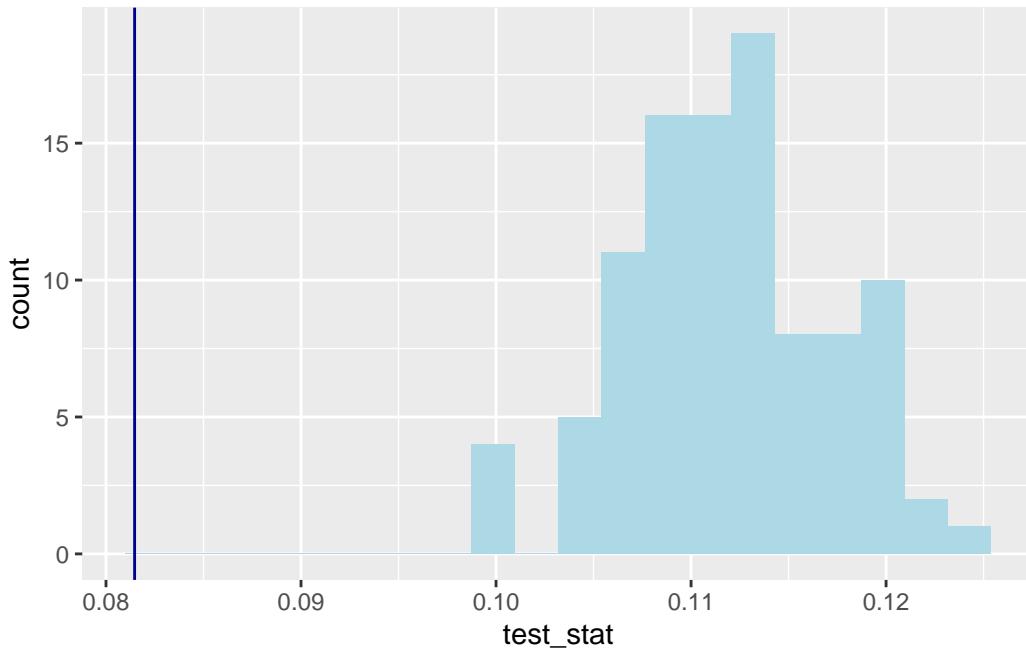
Model 1:

```
samp100_1 <- sample(nrow(yrep1), 100)  
samp100_1
```

```
[1] 42 737 241 327 44 335 744 743 583 670 220 379 229 947 695  
[16] 405 767 142 300 689 599 173 952 746 532 305 425 484 204 288  
[31] 92 370 772 575 864 750 146 501 215 253 97 295 243 298 569  
[46] 414 728 625 888 334 88 893 182 94 742 258 81 765 272 589  
[61] 429 672 52 129 970 457 139 357 398 358 769 990 726 603 541  
[76] 1000 475 526 165 813 59 997 533 107 988 535 321 100 462 360  
[91] 417 829 668 711 860 149 958 119 896 175
```

```
samples1 <- as_tibble(t(yrep1[samp100_1, ]))  
samples1 %>%  
  pivot_longer( everything(), names_to = "simulate", values_to = "Log_Weight" ) %>%  
  group_by(simulate) %>%  
  summarize(test_stat = mean(exp(Log_Weight) < 2.5)) %>%
```

```
ggplot(aes(x = test_stat)) +
  geom_histogram(bins = 20, fill = "lightblue") +
  geom_vline(xintercept = mean(ds$birthweight < 2.5), color = "darkblue")
```



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.

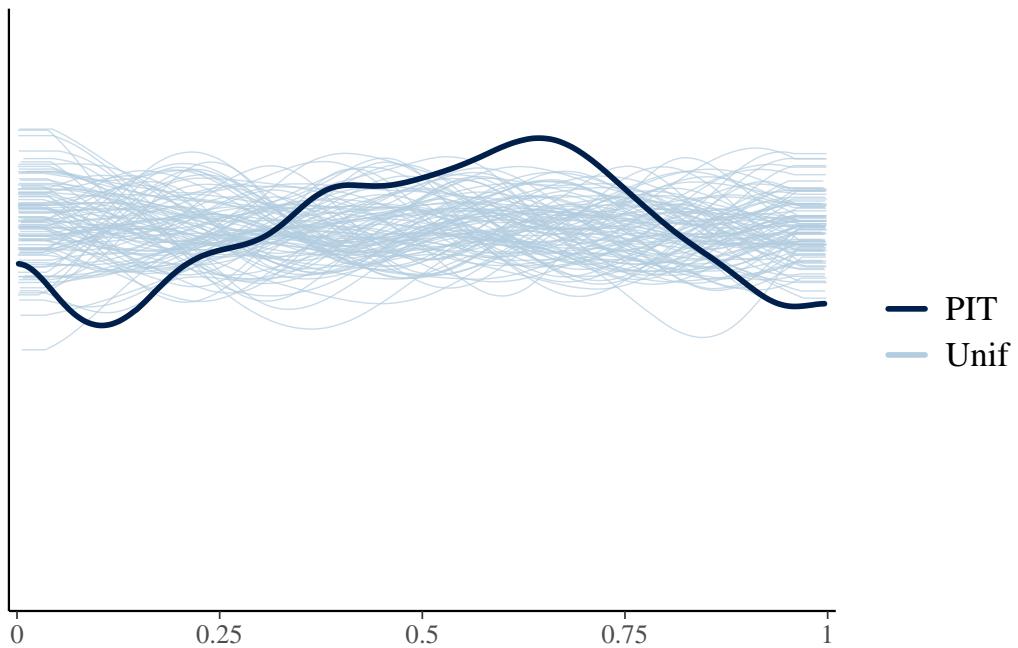
Comparing the two models tells us Model 2 is better:

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

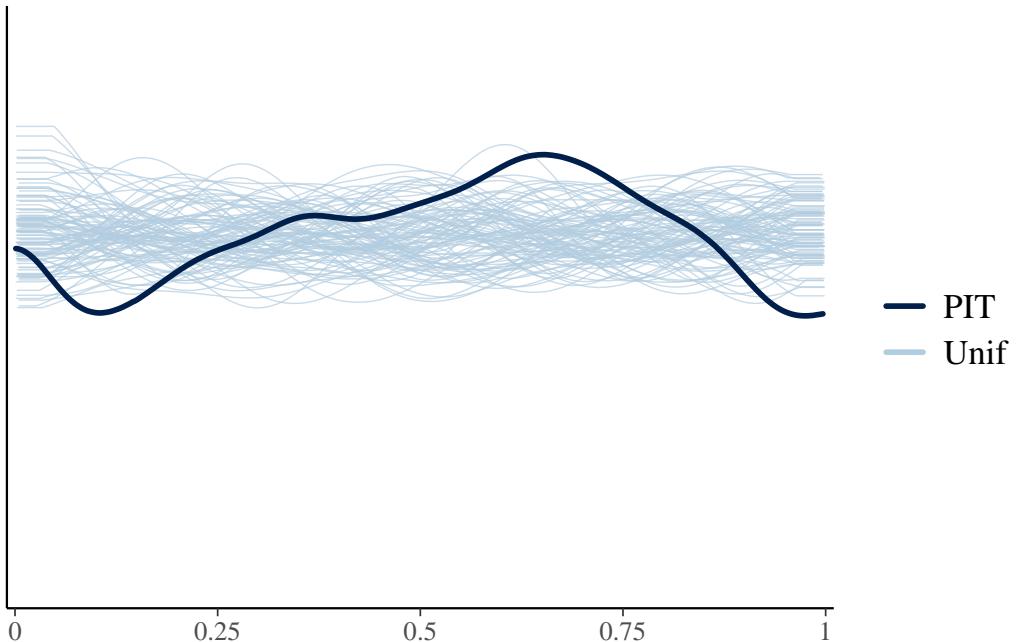
	elpd_diff	se_diff
model2	0.0	0.0
model1	-175.8	36.2

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.

Adding mother's age as covariate to the model.

```
stan_data <- list(N = nrow(ds),
                  log_weight = ds$log_weight,
                  log_gest = ds$log_gest_c,
                  preterm = ifelse(ds$preterm == "Y", 1, 0),
                  interaction = ds$log_gest_c*preterm,
                  age = ds$mager)

mod3 <- stan(data = stan_data,file = here("code/models/simple_weight_q8.stan"),iter = 1000)
```

```

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.001313 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 13.13 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration: 1 / 1000 [  0%] (Warmup)
Chain 1: Iteration: 100 / 1000 [ 10%] (Warmup)
Chain 1: Iteration: 200 / 1000 [ 20%] (Warmup)
Chain 1: Iteration: 300 / 1000 [ 30%] (Warmup)
Chain 1: Iteration: 400 / 1000 [ 40%] (Warmup)
Chain 1: Iteration: 500 / 1000 [ 50%] (Warmup)
Chain 1: Iteration: 501 / 1000 [ 50%] (Sampling)
Chain 1: Iteration: 600 / 1000 [ 60%] (Sampling)
Chain 1: Iteration: 700 / 1000 [ 70%] (Sampling)
Chain 1: Iteration: 800 / 1000 [ 80%] (Sampling)
Chain 1: Iteration: 900 / 1000 [ 90%] (Sampling)
Chain 1: Iteration: 1000 / 1000 [100%] (Sampling)
Chain 1:
Chain 1: Elapsed Time: 24.712 seconds (Warm-up)
Chain 1:                      43.058 seconds (Sampling)
Chain 1:                      67.77 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000639 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 6.39 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration: 1 / 1000 [  0%] (Warmup)
Chain 2: Iteration: 100 / 1000 [ 10%] (Warmup)
Chain 2: Iteration: 200 / 1000 [ 20%] (Warmup)
Chain 2: Iteration: 300 / 1000 [ 30%] (Warmup)
Chain 2: Iteration: 400 / 1000 [ 40%] (Warmup)
Chain 2: Iteration: 500 / 1000 [ 50%] (Warmup)
Chain 2: Iteration: 501 / 1000 [ 50%] (Sampling)
Chain 2: Iteration: 600 / 1000 [ 60%] (Sampling)
Chain 2: Iteration: 700 / 1000 [ 70%] (Sampling)
Chain 2: Iteration: 800 / 1000 [ 80%] (Sampling)

```

```

Chain 2: Iteration: 900 / 1000 [ 90%] (Sampling)
Chain 2: Iteration: 1000 / 1000 [100%] (Sampling)
Chain 2:
Chain 2: Elapsed Time: 44.977 seconds (Warm-up)
Chain 2:           53.401 seconds (Sampling)
Chain 2:           98.378 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).

Chain 3:
Chain 3: Gradient evaluation took 0.000685 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 6.85 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration: 1 / 1000 [  0%] (Warmup)
Chain 3: Iteration: 100 / 1000 [ 10%] (Warmup)
Chain 3: Iteration: 200 / 1000 [ 20%] (Warmup)
Chain 3: Iteration: 300 / 1000 [ 30%] (Warmup)
Chain 3: Iteration: 400 / 1000 [ 40%] (Warmup)
Chain 3: Iteration: 500 / 1000 [ 50%] (Warmup)
Chain 3: Iteration: 501 / 1000 [ 50%] (Sampling)
Chain 3: Iteration: 600 / 1000 [ 60%] (Sampling)
Chain 3: Iteration: 700 / 1000 [ 70%] (Sampling)
Chain 3: Iteration: 800 / 1000 [ 80%] (Sampling)
Chain 3: Iteration: 900 / 1000 [ 90%] (Sampling)
Chain 3: Iteration: 1000 / 1000 [100%] (Sampling)
Chain 3:
Chain 3: Elapsed Time: 39.174 seconds (Warm-up)
Chain 3:           45.523 seconds (Sampling)
Chain 3:           84.697 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).

Chain 4:
Chain 4: Gradient evaluation took 0.000655 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 6.55 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration: 1 / 1000 [  0%] (Warmup)
Chain 4: Iteration: 100 / 1000 [ 10%] (Warmup)
Chain 4: Iteration: 200 / 1000 [ 20%] (Warmup)

```

```

Chain 4: Iteration: 300 / 1000 [ 30%] (Warmup)
Chain 4: Iteration: 400 / 1000 [ 40%] (Warmup)
Chain 4: Iteration: 500 / 1000 [ 50%] (Warmup)
Chain 4: Iteration: 501 / 1000 [ 50%] (Sampling)
Chain 4: Iteration: 600 / 1000 [ 60%] (Sampling)
Chain 4: Iteration: 700 / 1000 [ 70%] (Sampling)
Chain 4: Iteration: 800 / 1000 [ 80%] (Sampling)
Chain 4: Iteration: 900 / 1000 [ 90%] (Sampling)
Chain 4: Iteration: 1000 / 1000 [100%] (Sampling)
Chain 4:
Chain 4:   Elapsed Time: 31.833 seconds (Warm-up)
Chain 4:           49.035 seconds (Sampling)
Chain 4:           80.868 seconds (Total)
Chain 4:

```

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

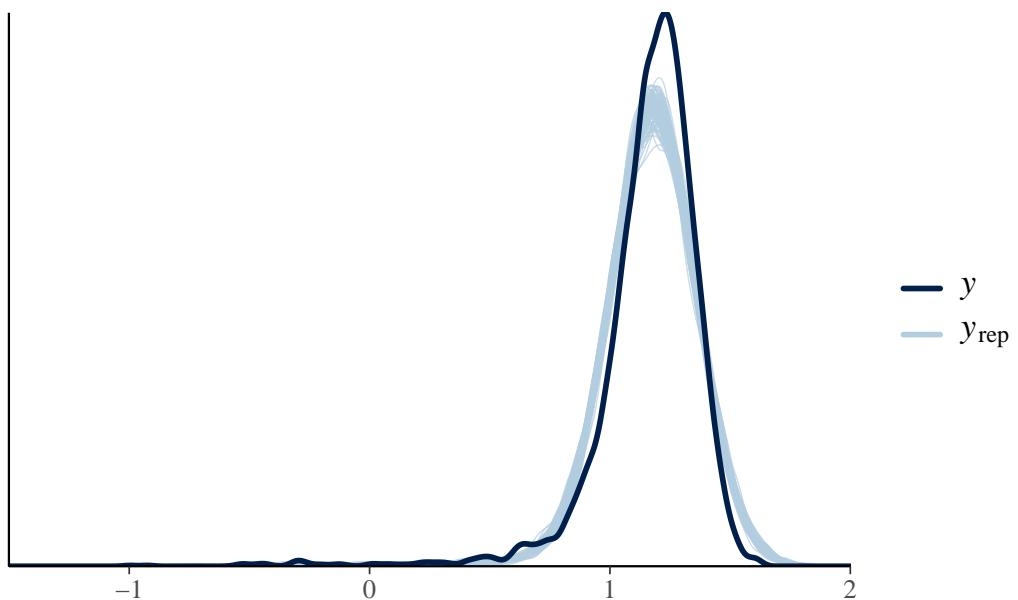
	mean	se_mean	sd	2.5%	25%
beta[1]	1.096836370	4.513563e-04	0.0130752067	1.071660621	1.087970125
beta[2]	0.102709090	8.283478e-05	0.0035119139	0.095662968	0.100369058
beta[3]	0.561859604	2.057254e-03	0.0622485677	0.440921574	0.519266917
beta[4]	0.197865269	4.422943e-04	0.0129336267	0.172965608	0.189173903
beta[5]	0.002516418	1.519570e-05	0.0004455417	0.001655092	0.002189443
sigma	0.160547116	6.541354e-05	0.0019870883	0.156676091	0.159195915
	50%	75%	97.5%	n_eff	Rhat
beta[1]	1.096889452	1.10610288	1.121422248	839.1856	1.007480
beta[2]	0.102716600	0.10513781	0.109474416	1797.4724	1.000943
beta[3]	0.561041715	0.60459695	0.688527149	915.5513	1.003016
beta[4]	0.197770659	0.20687603	0.223405544	855.1020	1.004142
beta[5]	0.002510789	0.00281973	0.003374534	859.6772	1.007356
sigma	0.160548712	0.16182020	0.164601763	922.7818	1.000615

```

y <- ds$log_weight
yrep3 <- extract(mod3)[["log_weight_rep"]]
samp100 <- sample(nrow(yrep3), 100)
ppc_dens_overlay(y, yrep3[samp100, ]) + ggtitle("Model 3: distribution of observed versus

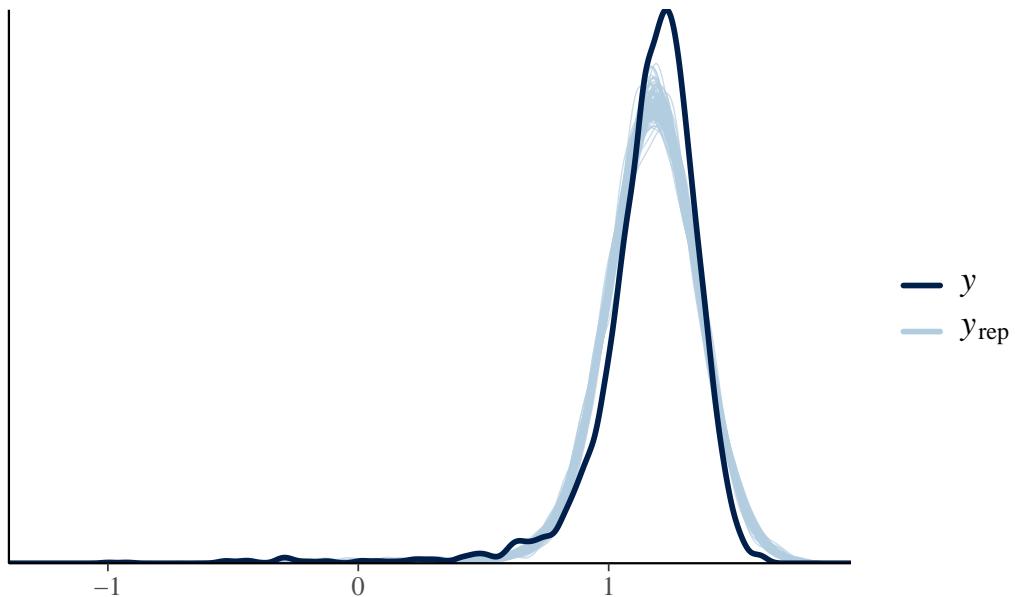
```

Model 3: distribution of observed versus predicted birthweights

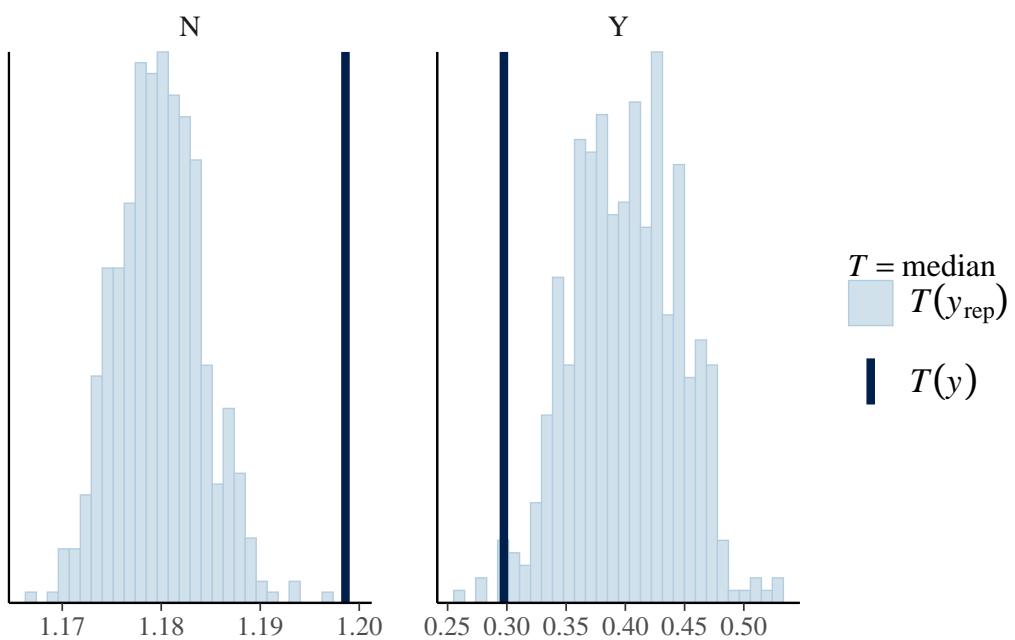


```
yrep2 <- extract(mod2)[["log_weight_rep"]]
samp100 <- sample(nrow(yrep2), 100)
ppc_dens_overlay(y, yrep2[samp100, ]) + ggtitle("Model 2:distribution of observed versus
```

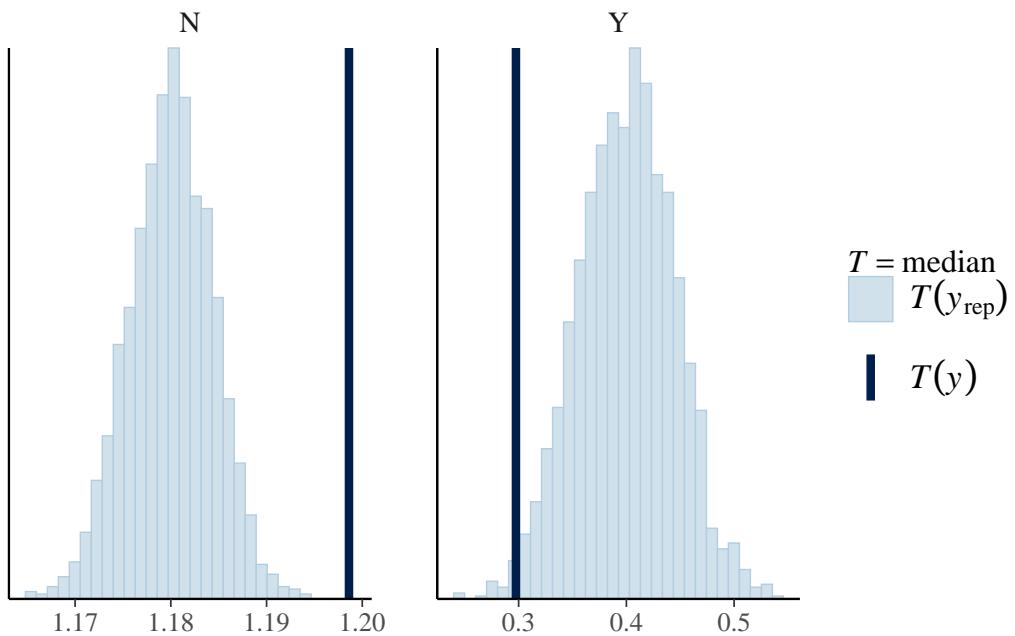
Model 2:distribution of observed versus predicted birthweights



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



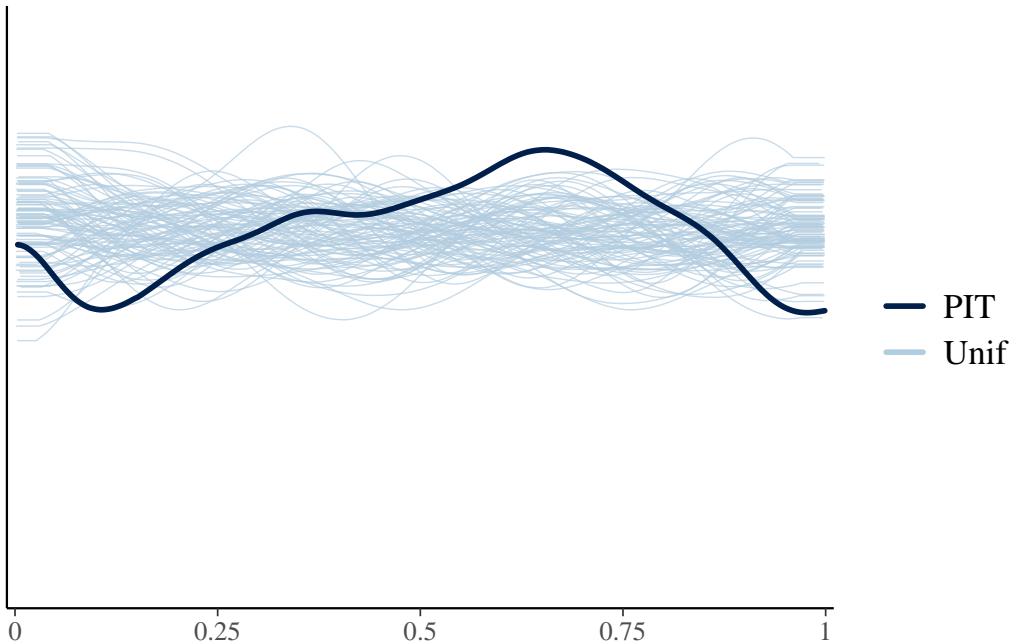
```
ppc_stat_grouped(ds$log_weight, yrep3, group = ds$preterm, stat = 'median')
```



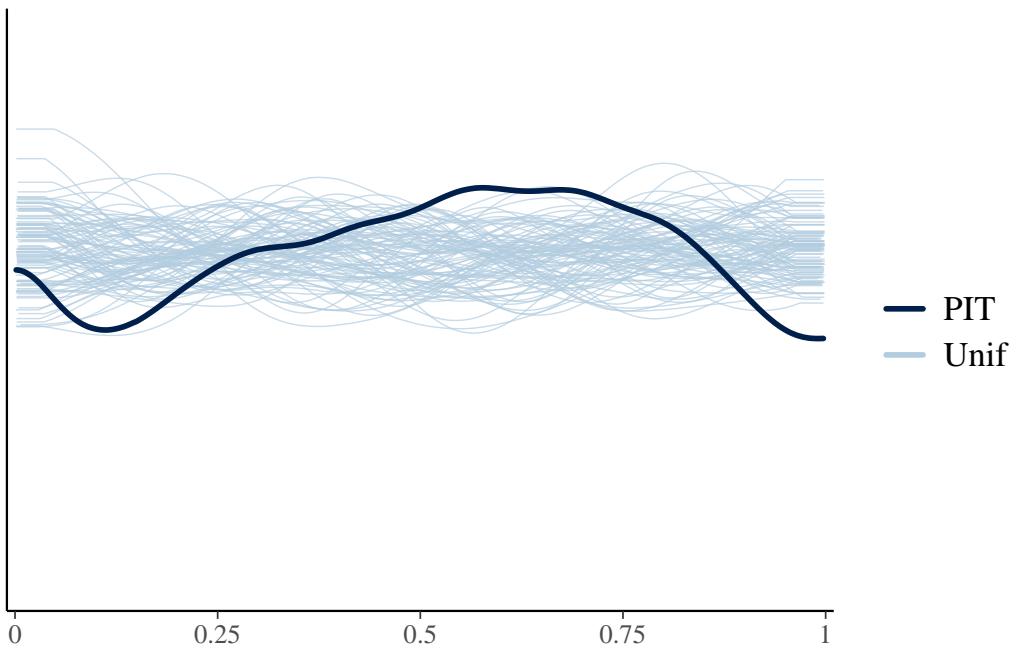
```
loglik3 <- extract(mod3)[["log_lik"]]
loo3 <- loo(loglik3, save_psis = TRUE)
loo_compare(loo2, loo3)
```

	elpd_diff	se_diff
model2	0.0	0.0
model1	-15.1	5.6

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



```
ppc_loo_pit_overlay(yrep = yrep3, y = y, lw = weights(loo3$psis_object))
```



Overall, from the distribution of observed versus predicted birthweights, test statistics(median) and LOO-PIT plot, we see that the two model is very similar. This is because the effect/coefficient(beta[5]) for mother's age is very small. This also reflected on loo_compare, that the elpd difference is only -15.1, indicating model 3 is slightly better than model 2 with mother's age added as covariate.