Week 6: Visualizing the Bayesian Workflow

27/02/23

Introduction

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

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

The data

Read it in, along with all our packages.

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

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

```
# A tibble: 6 x 8
  mager mracehisp meduc   bmi sex   combgest  dbwt ilive
  <dbl>     <dbl> <dbl> <dbl> <chr>    <dbl> <dbl> <chr>
1    16         2     2  23   M           39  3.18 Y
2    25         7     2  43.6 M           40  4.14 Y
```

```
3    27         2     3  19.5 F           41  3.18 Y
4    26         1     3  21.5 F           36  3.40 Y
5    28         7     2  40.6 F           34  2.71 Y
6    31         7     3  29.3 M           35  3.52 Y
```

Brief overview of variables:

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

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

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

Question 1

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

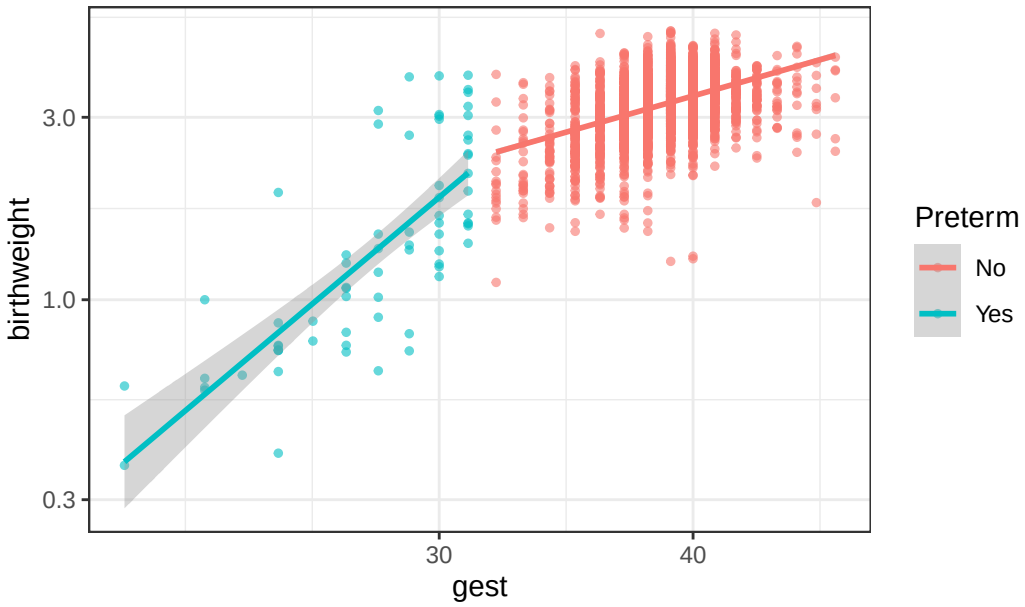
- Explain what your graph/ tables show
- Choose a graph type that's appropriate to the data type
- If you use `geom_smooth`, please also plot the underlying data

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

```
ds |>
  ggplot(aes(x = gest, y = birthweight, color = preterm)) +
  geom_point(size = 1, alpha = 0.6) +
  theme_bw() +
  ggtitle('Figure 1.1 Weight v gestational age') +
```

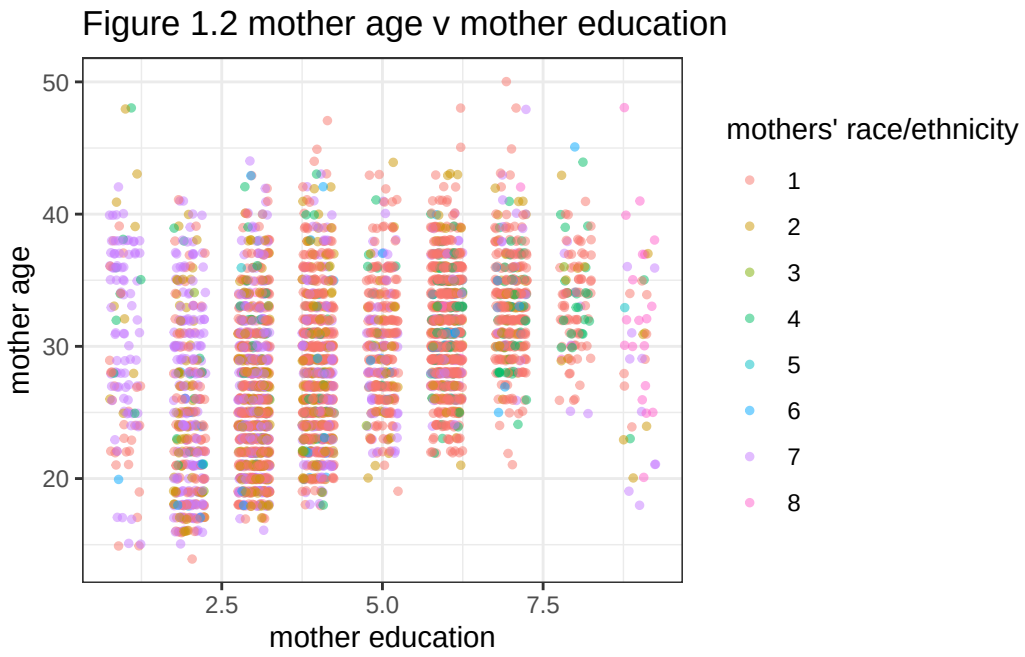
```
  labs(x = 'gest', y = 'birthweight', color = 'Preterm') +
  scale_color_discrete(labels = c('No', 'Yes')) +
  scale_x_log10() +
  scale_y_log10() +
  geom_smooth(method='lm', formula= y~x)
```

Figure 1.1 Weight v gestational age



- This is the replication of the class plot. We see from the plot that the relationship between birth weight and gest are steeper for preterm babies, and flatter for non-preterm babies.

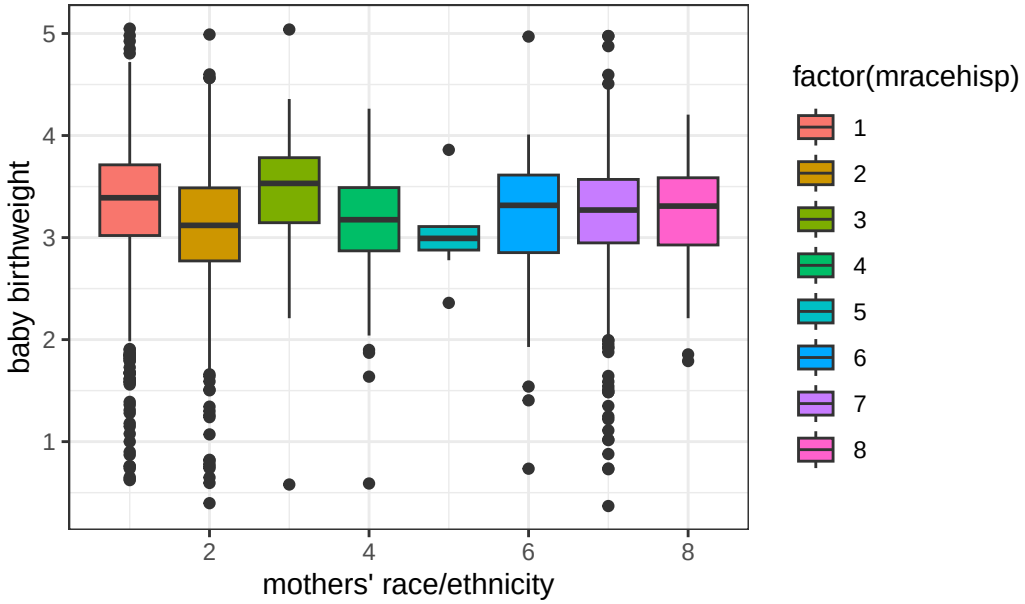
```
ds |>
  ggplot(aes(x = meduc, y = mager, color = as.factor(mracehisp))) +
  geom_point(size = 1, alpha = 0.5,position=position_jitter(h=0.1, w=0.25)) +
  theme_bw() +
  ggtitle('Figure 1.2 mother age v mother education') +
  labs(x = 'mother education', y = 'mother age', color = 'mothers\' race/ethnicity')
```



- This shows that mothers age and education has some level of collinearity, and their distributions are also impacted by mothers race/ethnicity.

```
ds |>
  ggplot(aes(x = mracehisp, y = birthweight, fill = factor(mracehisp))) +
  geom_boxplot() +
  theme_bw() +
  ggtitle('Figure 1.3 baby birthweight v mothers\' race/ethnicity') +
  labs(x = 'mothers\' race/ethnicity', y = 'baby birthweight')
```

Figure 1.3 baby birthweight v mothers' race/ethnicity



- While some race/ethnicity have similar babies' birth weight, for some other race/ethnicity the birth weight difference can be significant depends on the sample size.

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

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

- 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 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

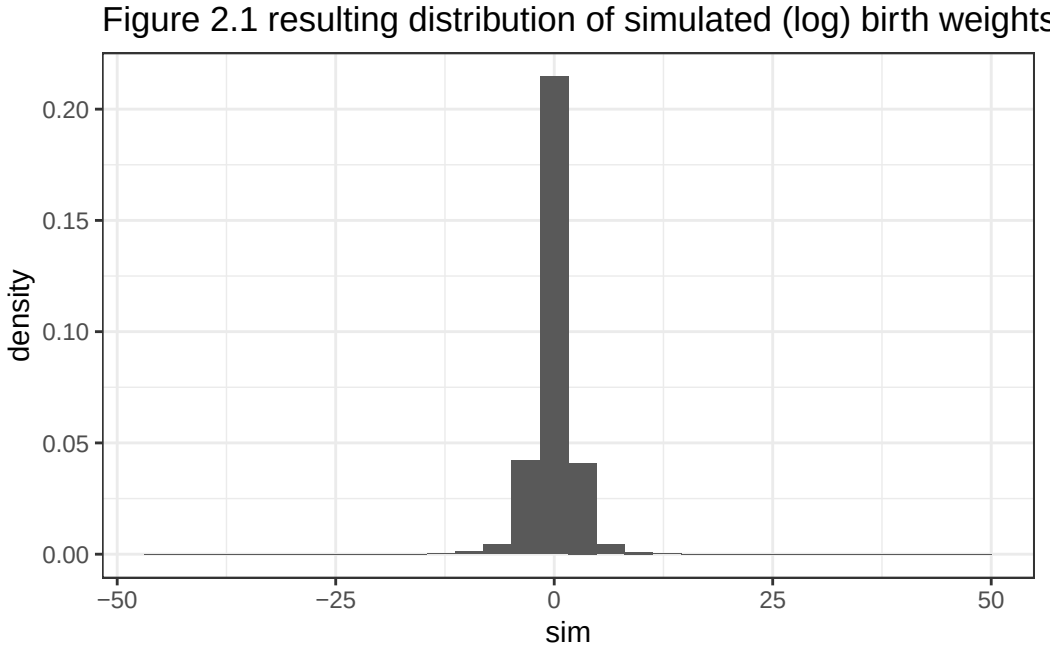
For Model 1, simulate values of βs and σ based on the priors above. Do 1000 simulations. Use these values to simulate (log) birth weights from the likelihood specified in Model 1, based on the set of observed gestational weights. **Remember the gestational weights should be centered and standardized**.

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

```
set.seed(314159)

sdd_gest = (log(ds$gest) - mean(log(ds$gest)))/sd(log(ds$gest))

b0 = rnorm(1000,0,1)
b1 = rnorm(1000,0,1)
s = abs(rnorm(1000,0,1))

sim = rep(0,1000*length(sdd_gest))
interitor = 1

for (i in 1:1000) {
  for (x in sdd_gest) {
    sim[interitor] <- rnorm(1,b0[i]+b1[i]*x,s[i]^2)
    interitor = interitor + 1
  }
```

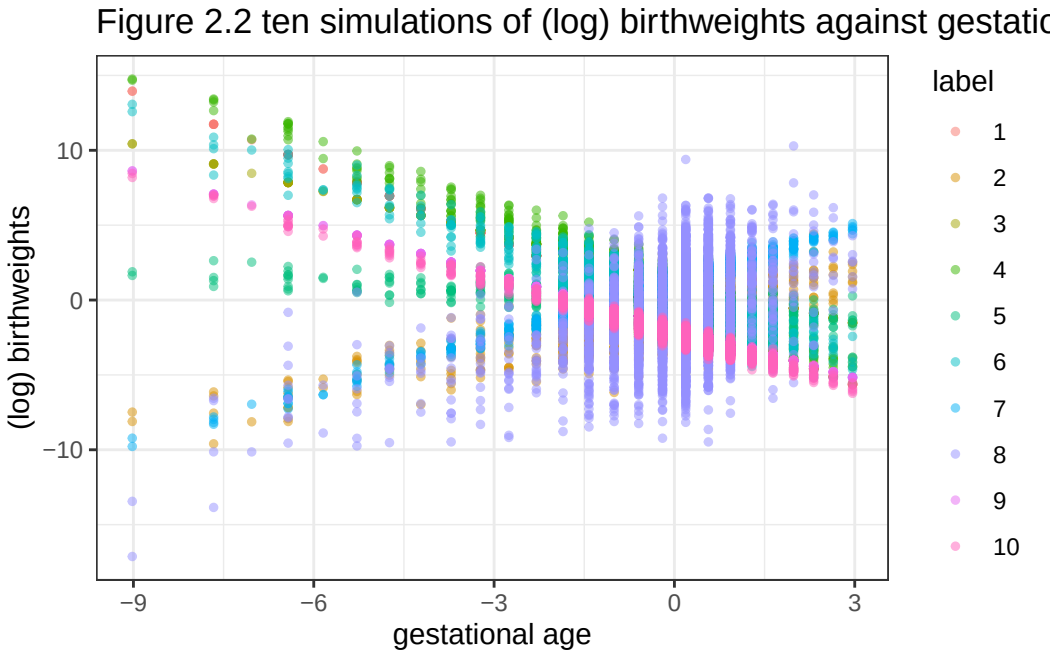
```
}
```

```
ggplot(data.frame(sim), aes(x = sim)) +
  geom_histogram(aes(y = after_stat(density))) +
  theme_bw() +
  ggtitle('Figure 2.1 resulting distribution of simulated (log) birth weights')
```




```
set.seed(314159)

sdd_gest = (log(ds$gest) - mean(log(ds$gest)))/sd(log(ds$gest))

b0 = rnorm(10,0,1)
b1 = rnorm(10,0,1)
s = abs(rnorm(10,0,1))

sim = rep(0,10*length(sdd_gest))
label = rep(0,10*length(sdd_gest))
iteritor = 1

for (i in 1:10) {
```

```
  for (x in sdd_gest) {
    sim[iteritor] <- rnorm(1,b0[i]+b1[i]*x,s[i]^2)
    label[iteritor] <- i
    iteritor = iteritor + 1
  }
}

sdd_gest = rep(sdd_gest, 10)

ggplot(data.frame(sim,sdd_gest, label), aes(x = sdd_gest, y = sim, color = as.factor(label
  geom_point(size = 1, alpha = 0.5) +
  theme_bw() +
  ggtitle('Figure 2.2 ten simulations of (log) birthweights against gestational age') +
  labs(x = 'gestational age', y = '(log) birthweights', color = 'label')
```

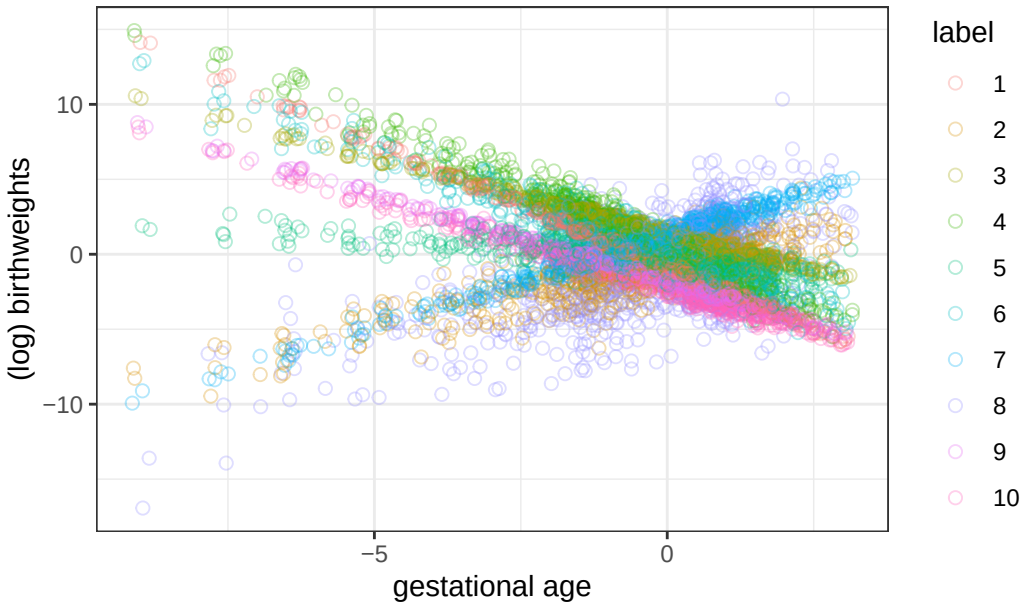


- I tried to fix the over plotting issue here, but was not very successful:

```
set.seed(314159)

sdd_gest = (log(ds$gest) - mean(log(ds$gest)))/sd(log(ds$gest))
```

```r
b0 = rnorm(10,0,1)
b1 = rnorm(10,0,1)
s = abs(rnorm(10,0,1))

sim = rep(0,10*length(sdd_gest))
label = rep(0,10*length(sdd_gest))
iteritor = 1

for (i in 1:10) {
  for (x in sdd_gest) {
    sim[iteritor] <- rnorm(1,b0[i]+b1[i]*x,s[i]^2)
    label[iteritor] <- i
    iteritor = iteritor + 1
  }
}

sdd_gest = rep(sdd_gest, 10)

samp <- sample(length(sdd_gest), 8000, prob = abs(sdd_gest-mean(sdd_gest))^2+0.1)

ggplot(data.frame(sim[samp],sdd_gest[samp],label[samp]), aes(x = sdd_gest[samp], y = sim[s
  geom_point(size = 2, position=position_jitter(h=0.2, w=0.2),
             shape = 21, alpha = 0.3) +
  theme_bw() +
  ggtitle('Figure 2.2 ten simulations of (log) birthweights against gestational age') +
  labs(x = 'gestational age', y = '(log) birthweights', color = 'label')
```

Figure 2.2 ten simulations of (log) birthweights against gestatic



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 0.000398 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 3.98 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: 0.823 seconds (Warm-up)
Chain 1:                0.67 seconds (Sampling)
Chain 1:                1.493 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000372 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 3.72 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: 0.802 seconds (Warm-up)
Chain 2:                0.744 seconds (Sampling)
Chain 2:                1.546 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000288 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.88 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: 0.799 seconds (Warm-up)
Chain 3:                0.683 seconds (Sampling)
Chain 3:                1.482 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000285 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.85 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 4: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 4: Iteration: 100 / 500 [ 20%]  (Warmup)
```

```
Chain 4: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 4: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 4: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 4: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 4: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 4: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 4: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 4: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 4: Iteration: 500 / 500 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.778 seconds (Warm-up)
Chain 4:                0.677 seconds (Sampling)
Chain 4:                1.455 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 1, give an estimate of the expected birth weight of a baby who was born at a gestational age of 37 weeks.

- using mean as the estimate for betas, the estimated log weight will be: 1.1625 + 0.1436*standardized[log(37)] = 1.077
- so the estimated birth weight will be exp(1.077) = 2.936 kg.

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

```
[1] 1.077005
```

```
exp(1.077005)
```

```
[1] 2.935873
```

Question 4

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

```
skimr::skim(ds)
```

Table 1: Data summary

Name	ds
Number of rows	3842
Number of columns	11
_____	
Column type frequency:	
character	3
numeric	8
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
sex	0	1	1	1	0	2	0
ilive	0	1	1	1	0	1	0
preterm	0	1	1	1	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
mager	0	1	28.93	5.84	14.00	25.00	29.00	33.00	50.00	
mracehisp	0	1	2.88	2.51	1.00	1.00	1.00	6.00	8.00	
meduc	0	1	4.41	1.81	1.00	3.00	4.00	6.00	9.00	
bmi	0	1	29.13	13.53	13.60	22.30	25.80	31.30	99.90	
gest	0	1	38.59	2.40	21.00	38.00	39.00	40.00	47.00	
birthweight	0	1	3.26	0.58	0.37	2.95	3.30	3.63	5.05	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
log_weight	0	1	1.16	0.22	-1.00	1.08	1.19	1.29	1.62	
log_gest_c	0	1	0.00	1.00	-9.02	-0.20	0.19	0.56	2.96	

```
ds <- ds |>
      mutate(preterm = ifelse(preterm == "N",0,1))
```

```
ds <- ds |>
      mutate(pt_log_g = preterm*log_gest_c)
```

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

Now fit the model

```
mod2q4 <- stan(data = stan_data,
               file = here("code/models/simple_weight_2.stan"),
               iter = 500,
               seed = 243)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.001393 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 13.93 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: 4.725 seconds (Warm-up)
Chain 1:                3.891 seconds (Sampling)
Chain 1:                8.616 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000947 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 9.47 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: 4.643 seconds (Warm-up)
Chain 2:                4.222 seconds (Sampling)
Chain 2:                8.865 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000945 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 9.45 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
```

```
Chain 3:
Chain 3: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 3: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 3: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 3: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 3: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 3: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 3: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 3: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 3: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 3: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 3: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 3: Iteration: 500 / 500 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 4.876 seconds (Warm-up)
Chain 3:                3.821 seconds (Sampling)
Chain 3:                8.697 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000946 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 9.46 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: 4.638 seconds (Warm-up)
Chain 4:                4.008 seconds (Sampling)
Chain 4:                8.646 seconds (Total)
Chain 4:
```

```
summary(mod2q4)$summary[c("beta[1]", "beta[2]", "beta[3]", "beta[4]", "sigma"),]
```

```
             mean      se_mean          sd       2.5%        25%       50%
beta[1] 1.1696329 8.021297e-05 0.002705139 1.16410383 1.16791913 1.1695478
beta[2] 0.1018545 1.111662e-04 0.003424916 0.09508969 0.09961365 0.1019319
beta[3] 0.5620695 3.406560e-03 0.062560942 0.43112646 0.52265217 0.5614275
beta[4] 0.1982641 6.964438e-04 0.012807594 0.17144797 0.18979854 0.1986269
sigma   0.1611971 8.785429e-05 0.001825790 0.15774991 0.15994557 0.1611909
              75%     97.5%     n_eff     Rhat
beta[1] 1.1714725 1.1748162 1137.3388 1.000638
beta[2] 0.1040358 0.1087724  949.1923 1.002232
beta[3] 0.6039584 0.6839901  337.2675 1.015352
beta[4] 0.2062635 0.2232005  338.1917 1.013325
sigma   0.1623667 0.1649513  431.8927 1.004553
```

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 results are similar.

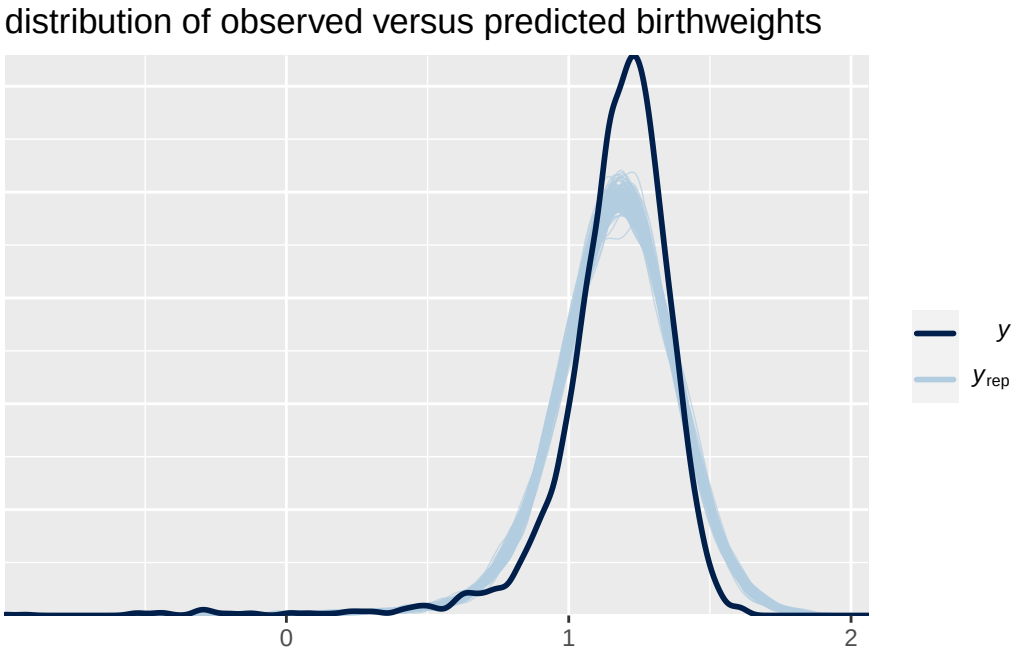
PPCs

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(mod2q4)[["log_weight_rep"]]
dim(yrep1)
```

```
[1] 1000 3842
```

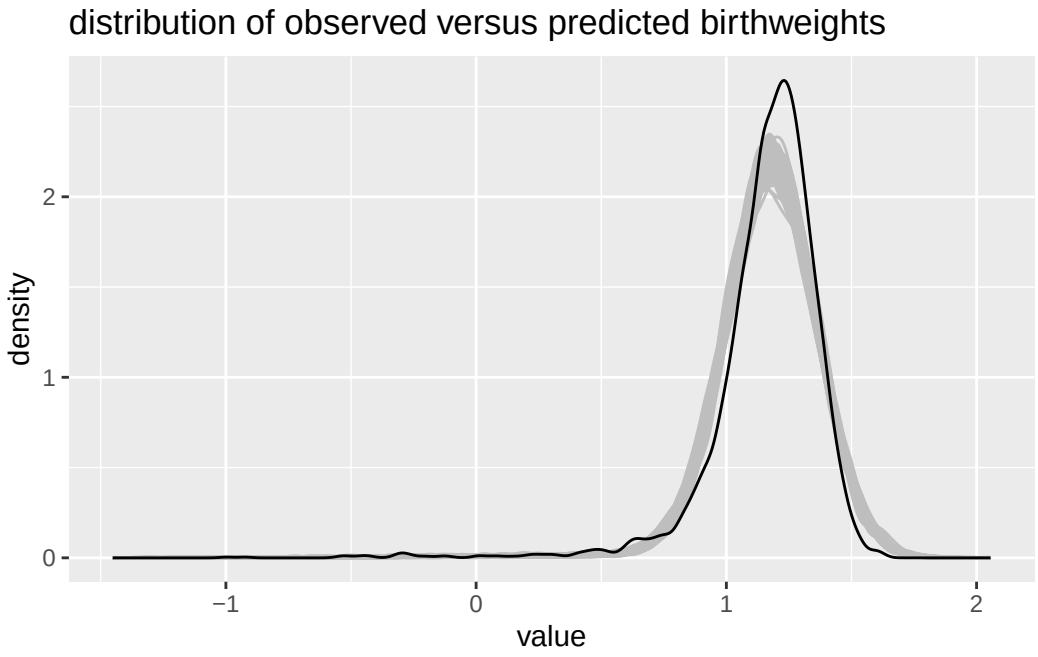
```
samp100 <- sample(nrow(yrep1), 100)
ppc_dens_overlay(y, yrep1[samp100, ])  + ggtitle("distribution of observed versus predicte
```



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`)

```
require(reshape2)
yrep2_m <- melt(yrep2[samp100,])

ggplot() +
  geom_density(data = yrep2_m, aes(x = value, group = iterations), color = 'grey') +
  geom_density(aes(x = y)) +
  ggtitle('distribution of observed versus predicted birthweights')
```

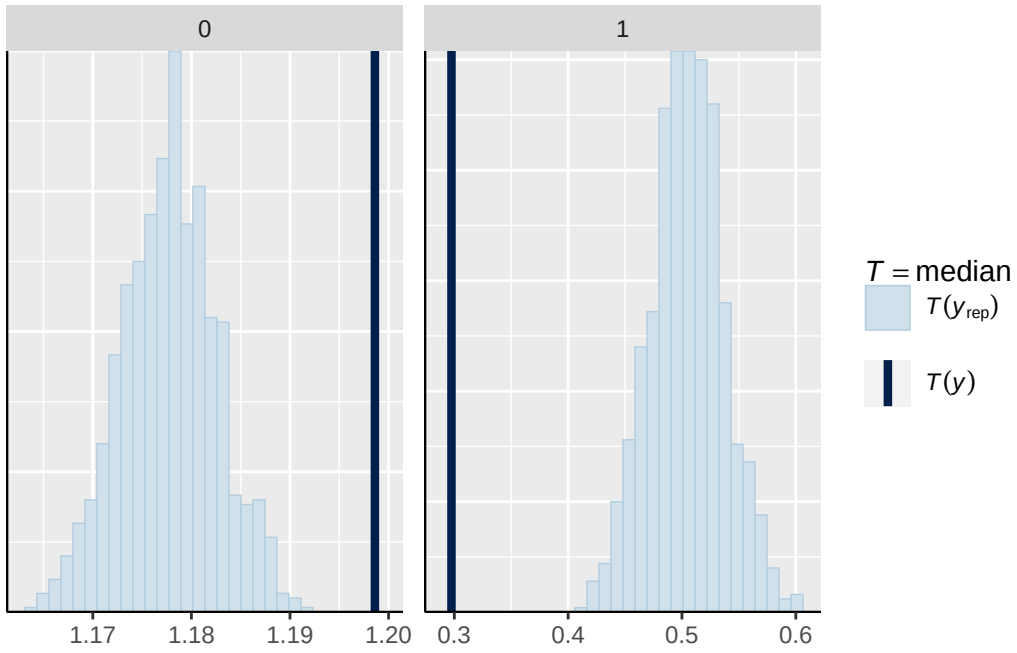


Test statistics

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

E.g. medians by prematurity for Model 1

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



Question 7

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

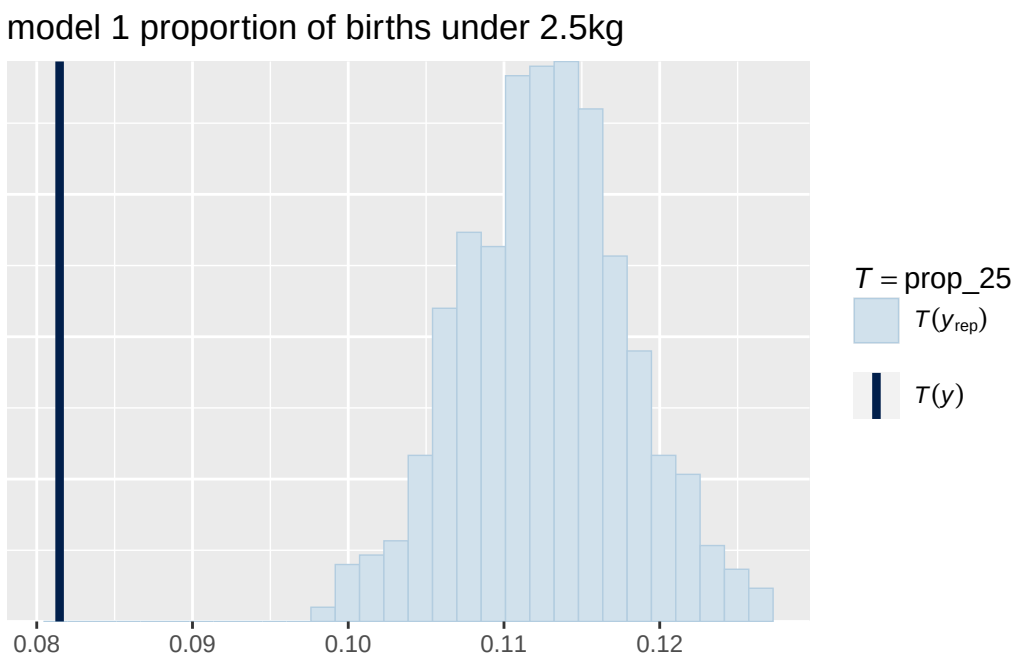
```
sum(y < log(2.5))/length(y)
```

```
[1] 0.08146799
```

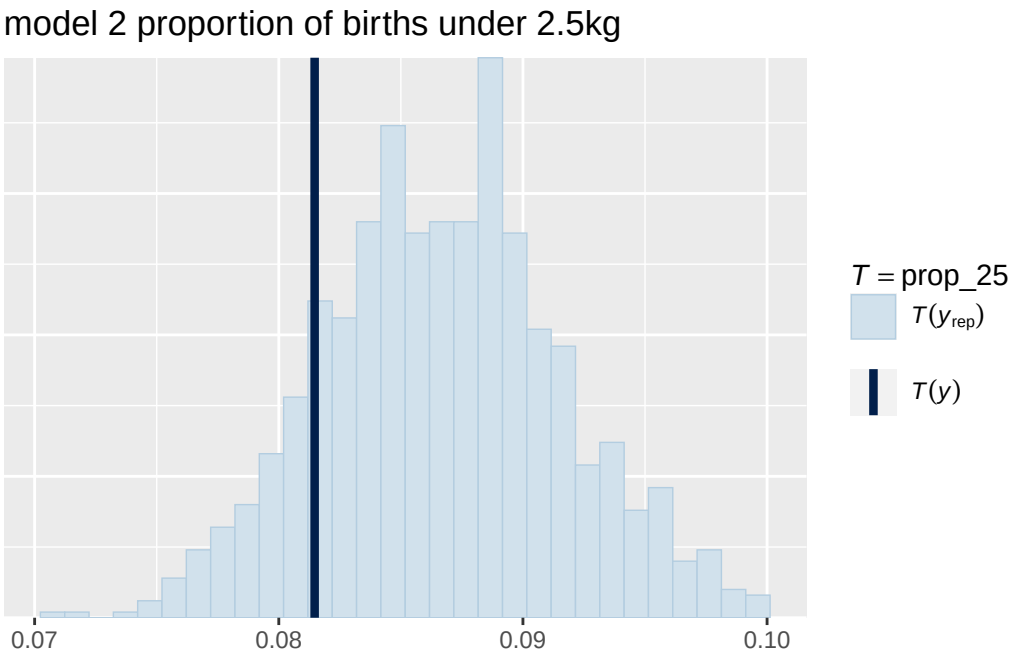
the test statistic for the data is about 0.081

```
prop_25 <- function(log_wt) {
  temp_prop <- sum(log_wt < log(2.5))/length(log_wt)
  return(temp_prop)
}

ppc_stat(ds$log_weight, yrep1, stat = prop_25) +
```

```
  ggtitle('model 1 proportion of births under 2.5kg')
```




```
ppc_stat(ds$log_weight, yrep2, stat = prop_25) +
  ggtitle('model 2 proportion of births under 2.5kg')
```



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(mod2q4)[["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 1000 by 3842 log-likelihood matrix

         Estimate    SE
elpd_loo   1552.7  69.8
p_loo        15.3   2.3
looic     -3105.4 139.7
------
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.
```

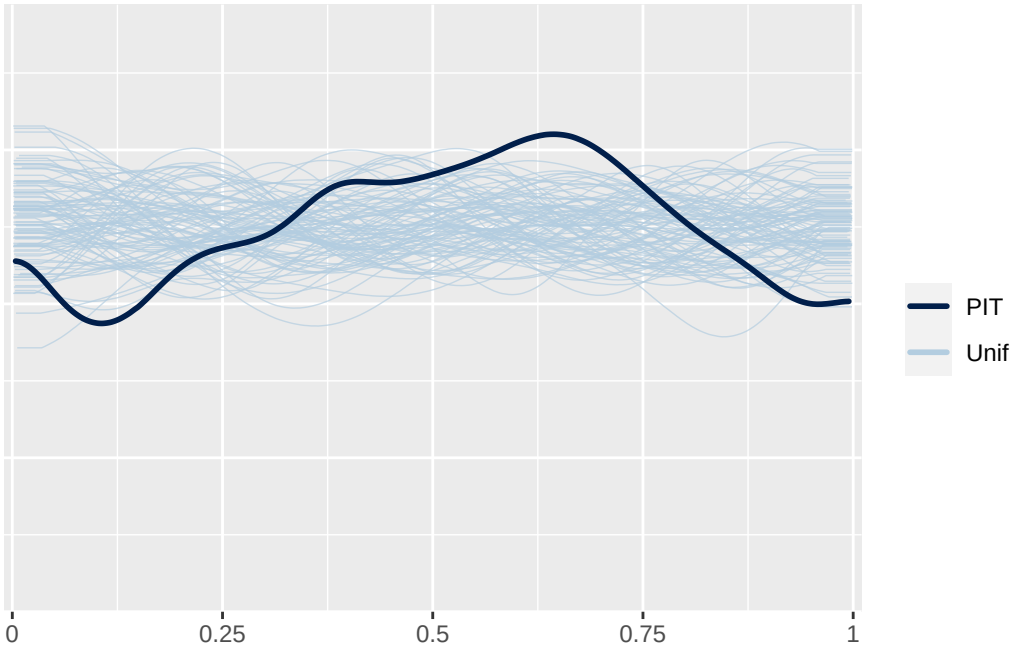
Comparing the two models tells us Model 2 is better:

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

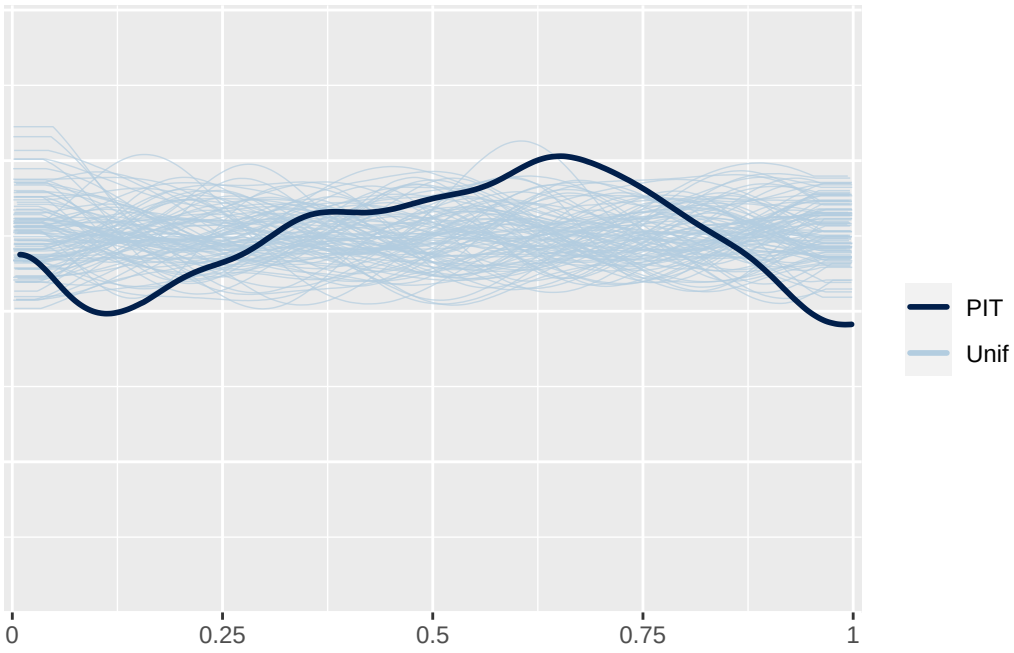
```
       elpd_diff se_diff
model2    0.0       0.0
model1 -175.6      36.3
```

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))
```



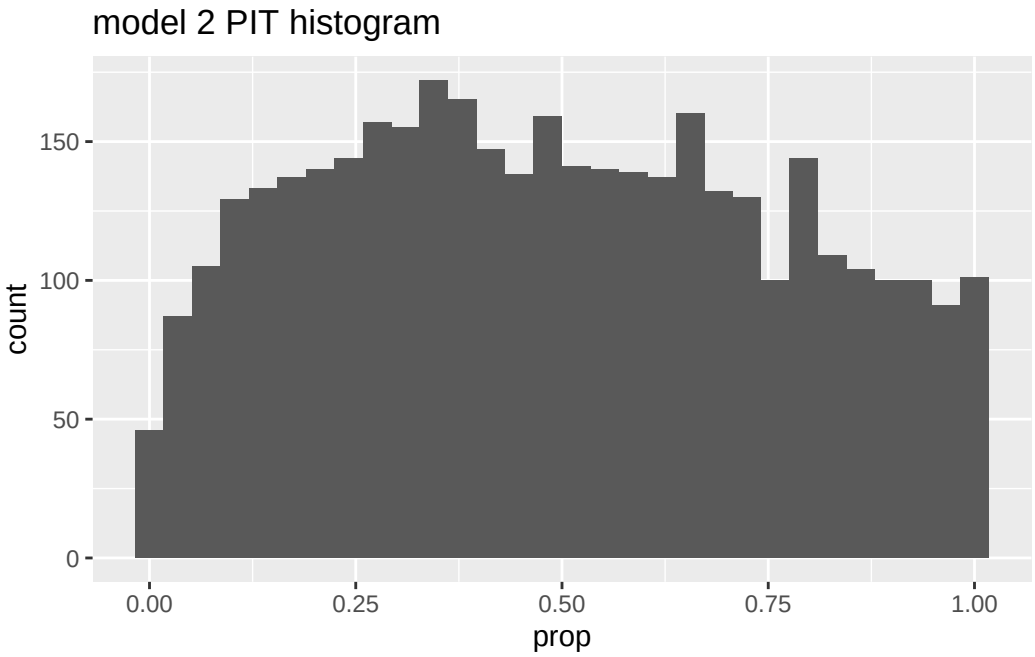

```
ppc_loo_pit_overlay(yrep = yrep2, y = y, lw = weights(loo2$psis_object))
```



Bonus question (not required)

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

```
prop <- rep(0,length(y))

for (i in 1:length(y)) {
  prop[i] = sum(y[i] <= yrep2[,i])/1000
}

ggplot(data = data.frame(prop), aes(x = prop)) +
  geom_histogram() +
  ggtitle('model 2 PIT histogram')
```



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.

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

- 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)
- w_i is mum's age

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

Now fit the model

```
mod3 <- stan(data = stan_data,
             file = here("code/models/simple_weight_3.stan"),
             iter = 500,
             seed = 243)
```

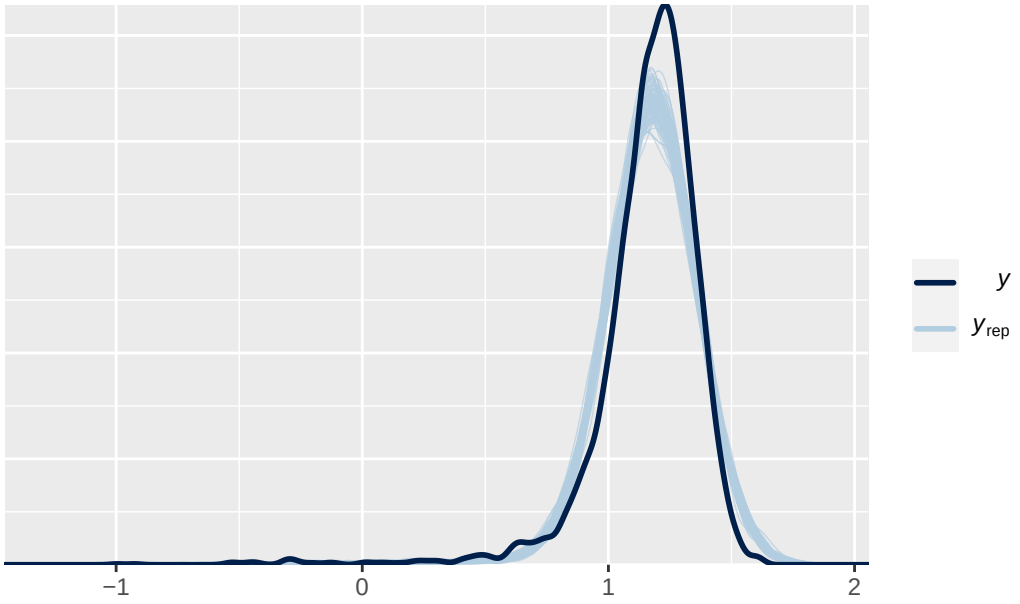
```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.002433 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 24.33 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:   1 / 500 [  0%]  (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: 9.816 seconds (Warm-up)
Chain 1:                95.465 seconds (Sampling)
```

```
Chain 1:                105.281 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.001289 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 12.89 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: 33.588 seconds (Warm-up)
Chain 2:                39.412 seconds (Sampling)
Chain 2:                73 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.001287 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 12.87 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: 25.757 seconds (Warm-up)
Chain 3:                31.154 seconds (Sampling)
Chain 3:                56.911 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.001337 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 13.37 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: 17.745 seconds (Warm-up)
Chain 4:                28.471 seconds (Sampling)
Chain 4:                46.216 seconds (Total)
Chain 4:
```

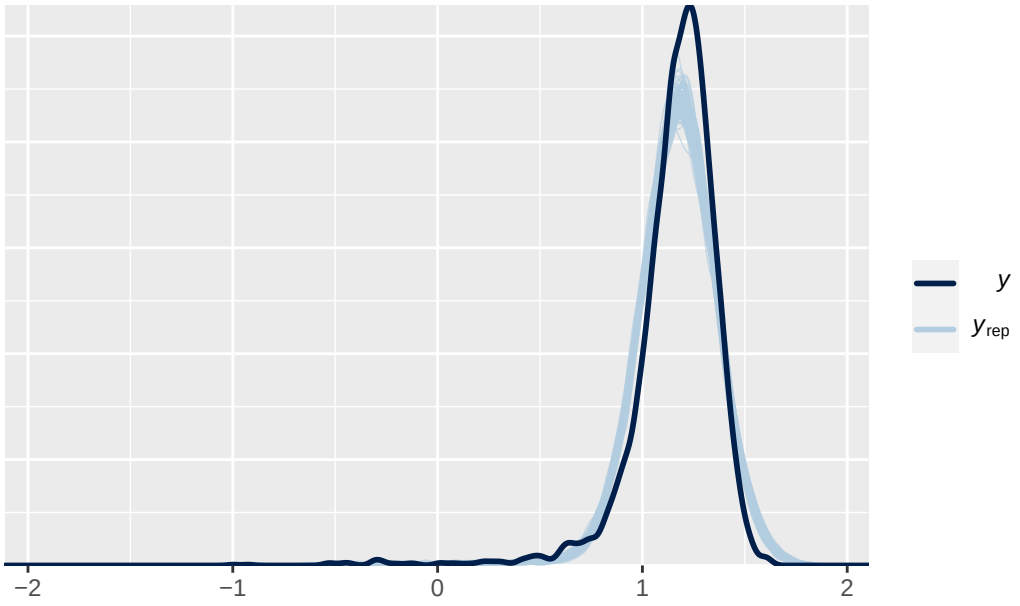
```
set.seed(1856)
yrep3 <- extract(mod3)[["log_weight_rep"]]
ppc_dens_overlay(y, yrep2[samp100, ])  + ggtitle("model 2 distribution of observed versus
```

model 2 distribution of observed versus predicted birthweights



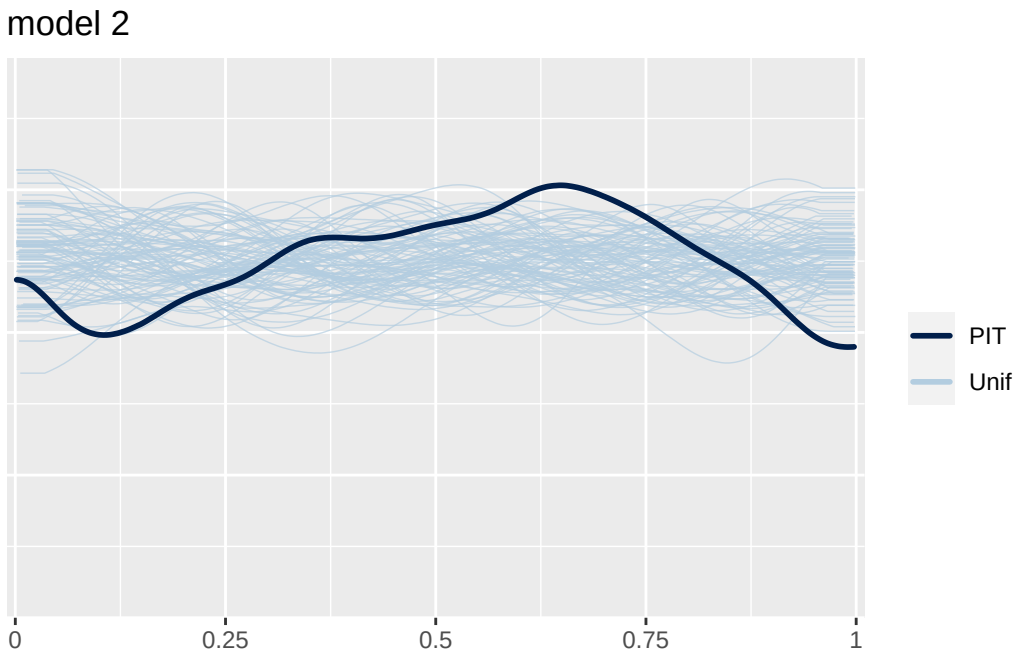

```
ppc_dens_overlay(y, yrep3[samp100, ])  + ggtitle("model 3 distribution of observed versus
```

model 3 distribution of observed versus predicted birthweights

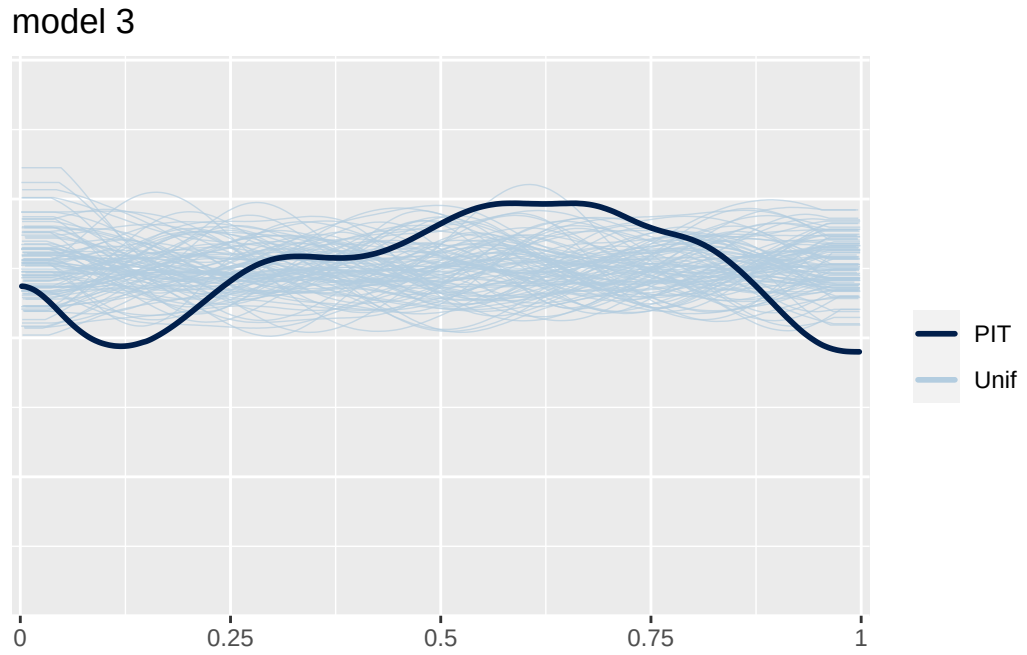


- The ppc density check does not show any visable improvement of model3 comparing to model 2.

```
ppc_loo_pit_overlay(yrep = yrep2, y = y, lw = weights(loo1$psis_object))+ ggtitle("model 2
```




```
ppc_loo_pit_overlay(yrep = yrep3, y = y, lw = weights(loo2$psis_object))+ ggtitle("model 3
```



- The ppc pit check shows that the pit for model 3 is flatter than model 2 and is more in line with uniform distributions, indicating some improvement of model 3.