Week 6: Visualizing the Bayesian Workflow

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, bmi<90)
```

Question 1

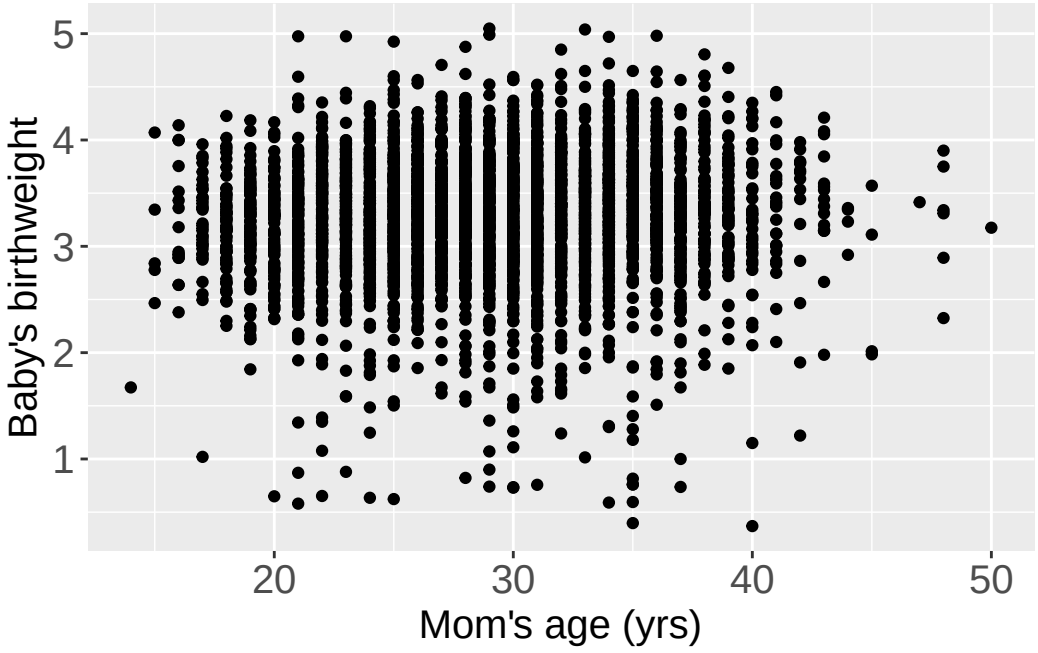
Use plots or tables to show three interesting observations about the data. Remember:

- 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.

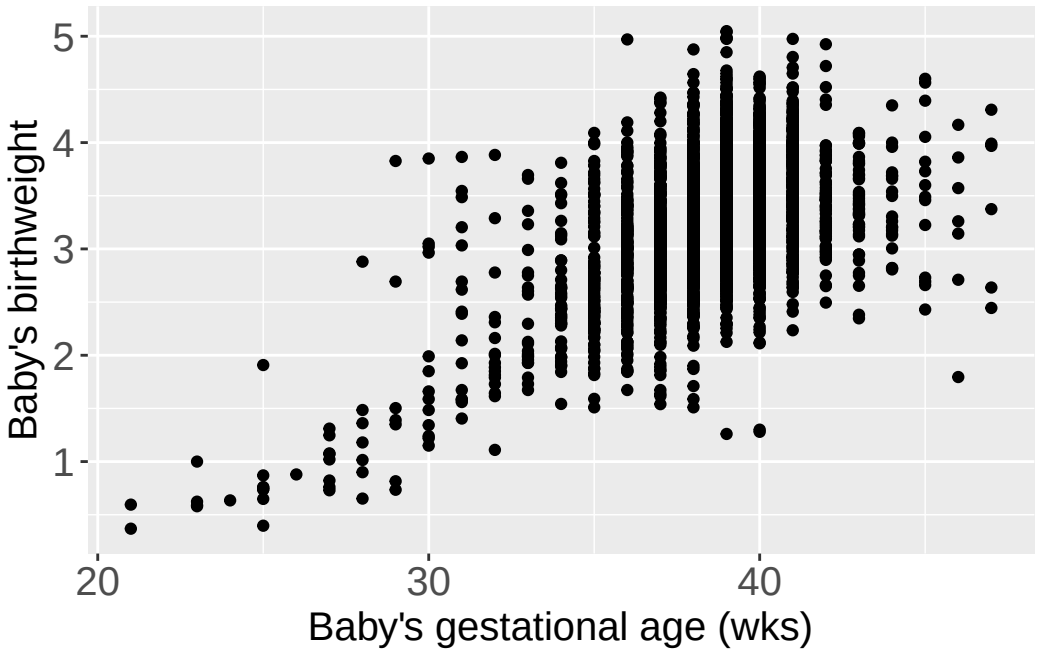
(1) Plot of mother's age vs baby's birthweight. The mother's age has a very weak relationship with the baby's birthweight. More spread of the baby's birthweight is seem for mother aged around 30.

```
theme_large_text <- theme(axis.text=element_text(size=15), axis.title=element_text(size=15

ds %>% ggplot(aes(x=mager, y=birthweight)) +
  geom_point(bins=50, color="black") +
  labs(x="Mom's age (yrs)", y="Baby's birthweight") +
  theme_large_text
```



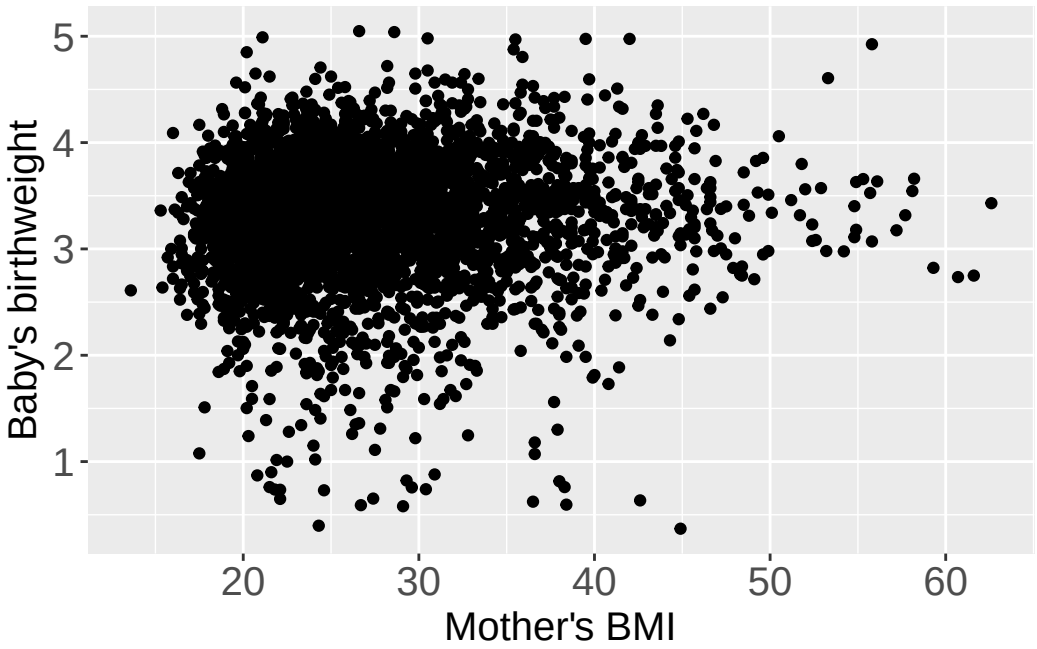
(2) Plot of baby's gestational age vs their weight. The gestational age shows a generally linear relationship with birthweight, with more spread in the term babies and the preterm babies.

```
ds %>% ggplot(aes(x=gest, y=birthweight)) +
  geom_point(bins=50, color="black") +
  labs(x="Baby's gestational age (wks)", y="Baby's birthweight") +
  theme_large_text
```



(3) Plot of mother's body mass index (BMI) vs baby's birthweight. The mother's BMI shows weak relation w.r.t. the baby's birthweights.

```
ds %>% ggplot(aes(x=bmi, y=birthweight)) +
  geom_point(bins=50, color="black") +
  labs(x="Mother's BMI", y="Baby's birthweight") +
  theme_large_text
```



The model

As in lecture, we will look at two candidate models

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

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

Model 2 has an interaction term between gestation and prematurity

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

- y_i is weight in kg
- x_i is gestational age in weeks, CENTERED AND STANDARDIZED
- z_i is preterm (0 or 1, if gestational age is less than 32 weeks)

Prior predictive checks

Let's put some weakly informative priors on all parameters i.e. for the βs

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

and for σ

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

where the plus means positive values only i.e. Half Normal.

Let's check to see what the resulting distribution of birth weights look like given Model 1 and the priors specified above, assuming we had no data on birth weight (but observations of gestational age).

Question 2

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

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

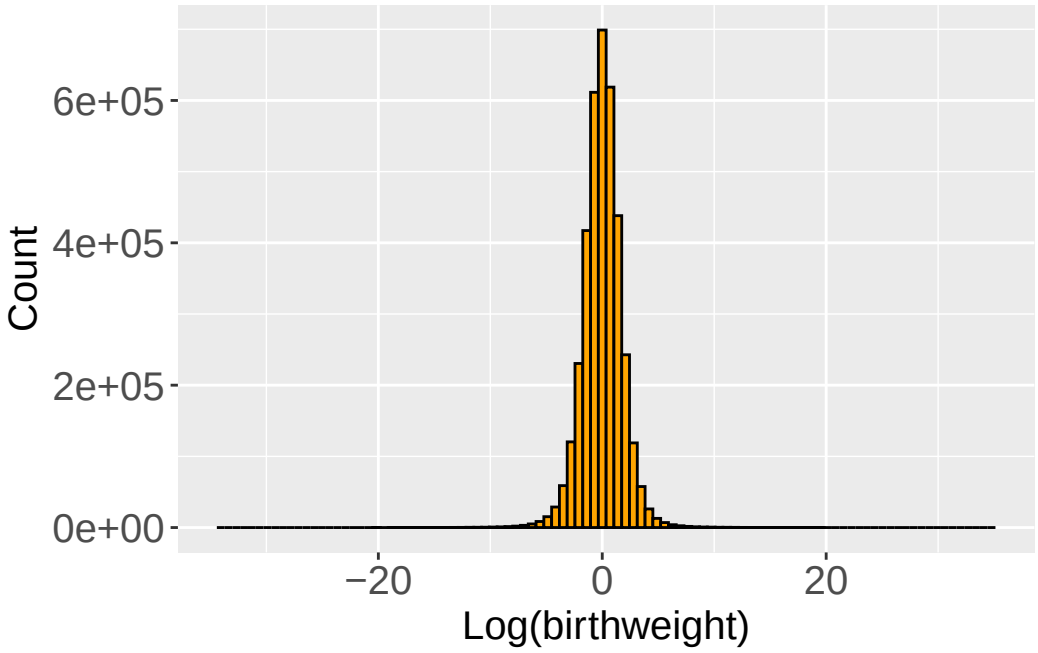
Run simulation of the linear model

```
nsims <- 1000
sigma <- abs(rnorm(nsims, 0, 1))
beta0 <- rnorm(nsims, 0, 1)
beta1 <- rnorm(nsims, 0, 1)

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

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

Simulated birthweights on logscale

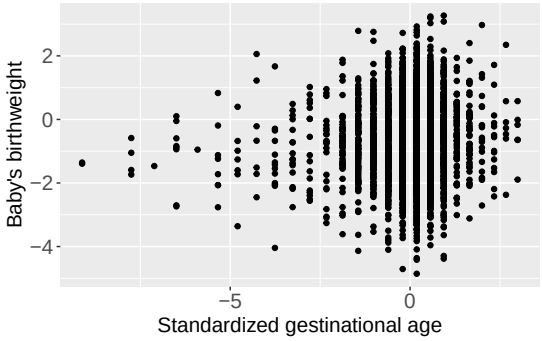
```
dsl <- dsims %>%
  pivot_longer(`1`:`1000`, names_to = "sim", values_to = "sim_weight")

dsl %>%
```

```
  ggplot(aes(x=sim_weight)) +
  geom_histogram(bins=100, color="black", fill="orange") + labs(x='Log(birthweight)', y='C
```



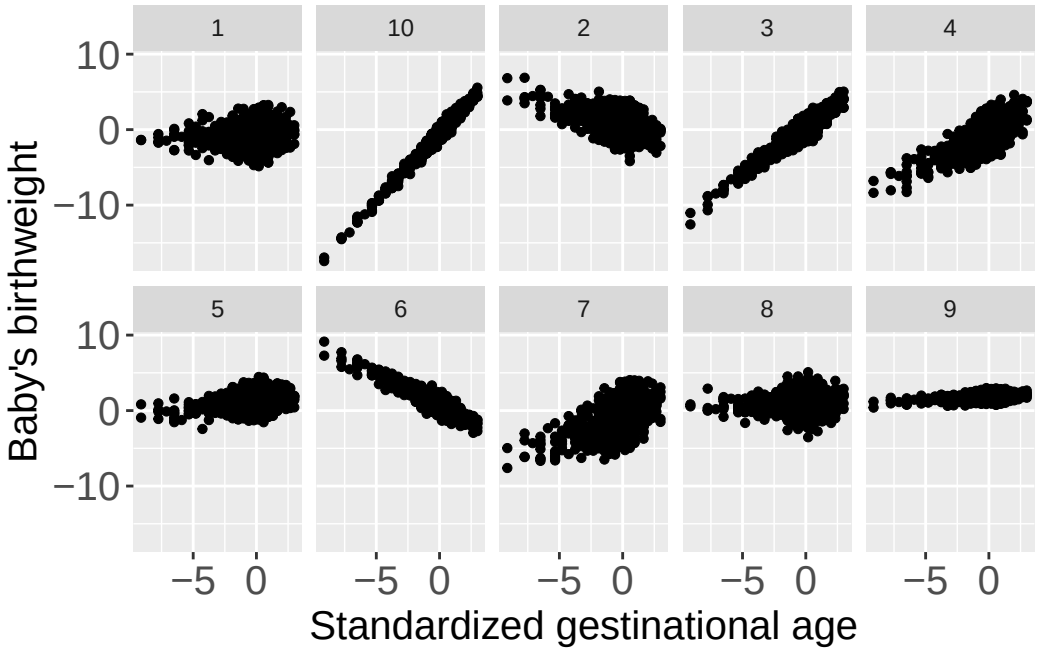
Simulated birthweights vs gestational age for 1 simulation

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

dsl %>%
  ggplot(aes(x=log_gest_centered, y=sim_weight)) +
  geom_point(stroke=0.5) +
  labs(x="Standardized gestinational age", y="Baby's birthweight") +
  theme_large_text
```



Simulated birthweights vs gestational age for 10 simulations

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

dsl %>%
  ggplot(aes(x=log_gest_centered, y=sim_weight)) +
  geom_point(stroke=0.2) +
  labs(x="Standardized gestational age", y="Baby's birthweight") +
  theme_large_text + facet_wrap(~sim, nrow=2)
```



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("./Lab6/simple_weight.stan"),
             iter = 500,
             seed = 243)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000504 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 5.04 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: 0.697 seconds (Warm-up)
Chain 1:                0.597 seconds (Sampling)
Chain 1:                1.294 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000323 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 3.23 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: 0.703 seconds (Warm-up)
Chain 2:                0.6 seconds (Sampling)
Chain 2:                1.303 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000206 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.06 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: 0.644 seconds (Warm-up)
Chain 3:                0.609 seconds (Sampling)
Chain 3:                1.253 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.0002 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2 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: 0.686 seconds (Warm-up)
Chain 4:                0.584 seconds (Sampling)
Chain 4:                1.27 seconds (Total)
Chain 4:
```

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

```
            mean      se_mean          sd      2.5%       25%       50%
beta[1] 1.1633773 7.543662e-05 0.002773092 1.1579418 1.1615077 1.1632311
```

```
beta[2] 0.1420249 8.659012e-05 0.002770712 0.1368342 0.1401326 0.1420730
sigma   0.1684490 1.440100e-04 0.002138198 0.1638366 0.1670508 0.1683676
              75%     97.5%     n_eff      Rhat
beta[1] 1.1653385 1.1685818 1351.3382 0.9969935
beta[2] 0.1438496 0.1473567 1023.8729 1.0026420
sigma   0.1698146 0.1725994  220.4502 1.0201324
```

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.

```
# Convert to standardized value
gestage <- (log(37) - mean(log(ds$gest)))/sd(log(ds$gest))
# Extract fit parameters from the stan model
mod1_params <- summary(mod1)$summary[c("beta[1]", "beta[2]", "sigma"),]
mod1_params
```

```
             mean      se_mean          sd      2.5%       25%       50%
beta[1] 1.1633773 7.543662e-05 0.002773092 1.1579418 1.1615077 1.1632311
beta[2] 0.1420249 8.659012e-05 0.002770712 0.1368342 0.1401326 0.1420730
sigma   0.1684490 1.440100e-04 0.002138198 0.1638366 0.1670508 0.1683676
              75%     97.5%     n_eff      Rhat
beta[1] 1.1653385 1.1685818 1351.3382 0.9969935
beta[2] 0.1438496 0.1473567 1023.8729 1.0026420
sigma   0.1698146 0.1725994  220.4502 1.0201324
```

```
b1 = mod1_params['beta[1]', 'mean']
b2 = mod1_params['beta[2]', 'mean']
bw = exp(b1 + b2 * gestage)

sprintf("The predicted birthweight at a gestinational age of 37 weeks is %f", bw)
```

```
[1] "The predicted birthweight at a gestinational age of 37 weeks is 2.936982"
```

Question 4

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

```r
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_2 <- list(N = nrow(ds),
                    log_weight = ds$log_weight,
                    log_gest = ds$log_gest_c,
                    z = ifelse(ds$preterm == "Y", 1, 0))
```

Now fit the model

```r
mod2 <- stan(data = stan_data_2,
             file = here("./Lab6/mod2_weight.stan"),
             iter = 500,
             seed = 243)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.001249 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 12.49 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 1: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 1: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 1: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 1: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 1: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 1: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 1: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 1: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 1: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 1: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 1: Iteration: 500 / 500 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 5.596 seconds (Warm-up)
Chain 1:                3.957 seconds (Sampling)
Chain 1:                9.553 seconds (Total)
Chain 1:
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000527 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 5.27 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: 4.354 seconds (Warm-up)
Chain 2:                2.244 seconds (Sampling)
Chain 2:                6.598 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000461 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 4.61 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 3: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 3: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 3: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 3: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 3: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 3: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 3: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 3: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 3: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 3: Iteration: 450 / 500 [ 90%]  (Sampling)
```

```
Chain 3: Iteration: 500 / 500 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 3.201 seconds (Warm-up)
Chain 3:                2.368 seconds (Sampling)
Chain 3:                5.569 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.00066 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 6.6 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 4: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 4: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 4: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 4: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 4: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 4: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 4: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 4: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 4: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 4: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 4: Iteration: 500 / 500 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 2.574 seconds (Warm-up)
Chain 4:                2.071 seconds (Sampling)
Chain 4:                4.645 seconds (Total)
Chain 4:
```

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

```
            mean      se_mean          sd       2.5%        25%       50%
beta[1] 1.1703015 6.250814e-05 0.002549552 1.16521743 1.16853182 1.1704160
beta[2] 0.1003671 1.107223e-04 0.003621365 0.09326558 0.09807501 0.1004429
beta[3] 0.5413710 3.200004e-03 0.064622188 0.41736298 0.49695937 0.5392622
beta[4] 0.1936998 6.657405e-04 0.013341416 0.16827237 0.18502010 0.1929840
sigma   0.1609163 8.090727e-05 0.001959520 0.15702522 0.15963065 0.1608958
             75%     97.5%     n_eff      Rhat
```

```
beta[1] 1.1720218 1.1751571 1663.6218 0.9974635
beta[2] 0.1029331 0.1071365 1069.7299 0.9980882
beta[3] 0.5851898 0.6675624  407.8142 1.0078418
beta[4] 0.2027231 0.2198585  401.6001 1.0072110
sigma   0.1621782 0.1651113  586.5763 0.9998487
```

Question 5

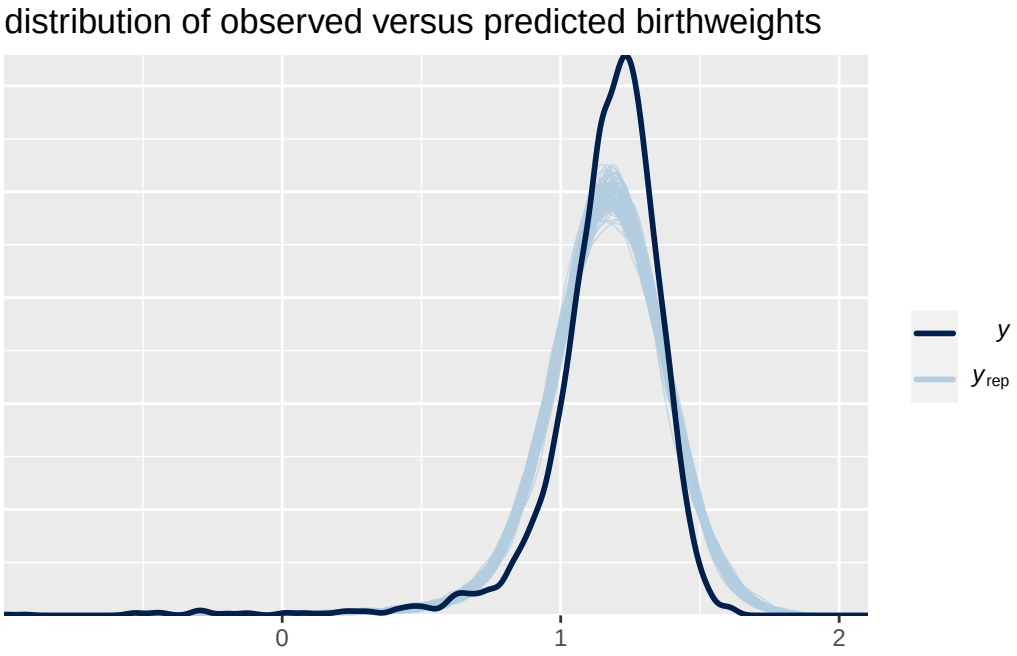
For reference I have uploaded some model 2 results. Check your results are similar.

```
## Loading this model can work but later results in some dimension mismatch with model 1 u

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

PPCs

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

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

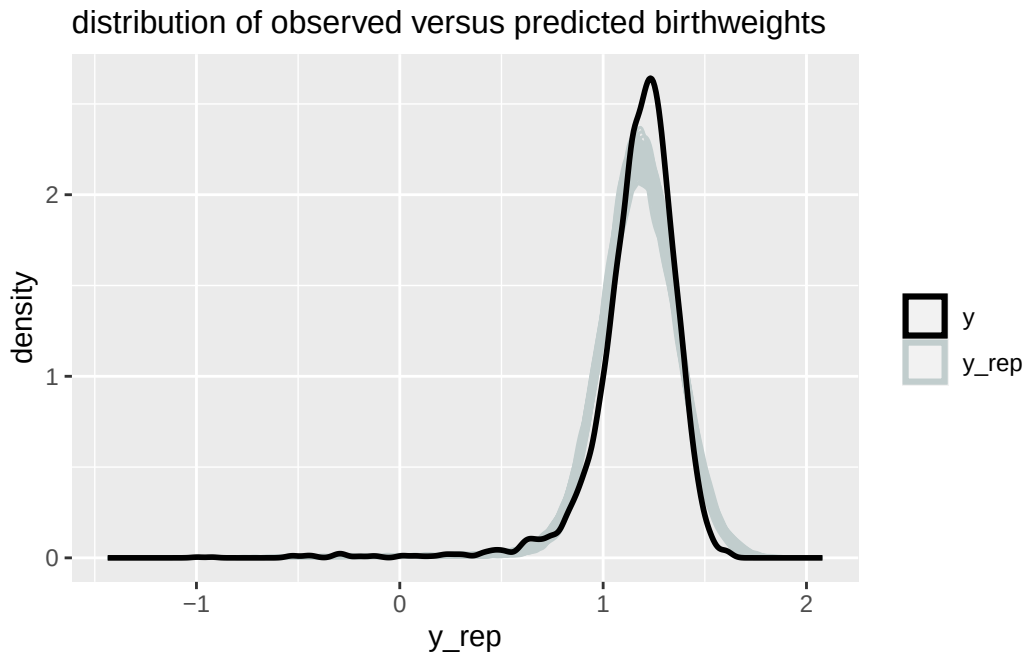


Question 6

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`)

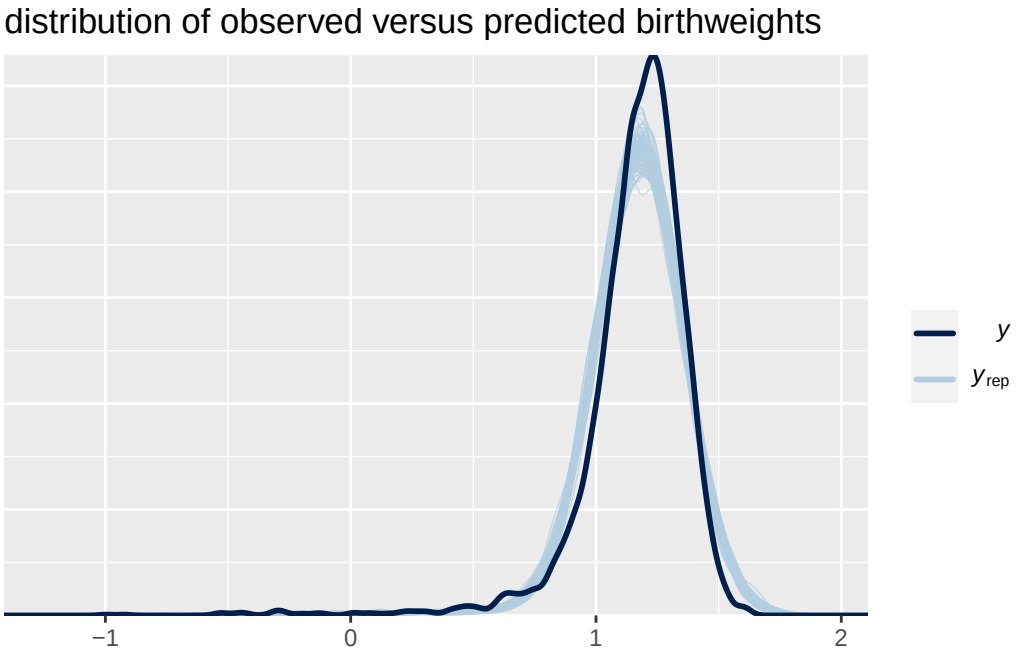
```
# Randomly sample 100 replicates from simulated data
samp100 <- sample(nrow(yrep2), 100)
rownames(yrep2) <- 1:nrow(yrep2)
# Transpose yrep2 (replicates from model 2)
d2 <- as_tibble(t(yrep2))
d2 <- d2 %>% bind_cols(i = 1:nrow(ds),
            log_weight_obs = log(ds$birthweight))

# Change into long format
drep2 <- d2 %>%
  pivot_longer(-(i:log_weight_obs),
               names_to = "sim", values_to ="y_rep")

# Take only 100 samples for plotting
drep2<- drep2 %>% filter(sim %in% samp100)
drep2 %>% ggplot(aes(y_rep, group = sim)) +
  geom_density(alpha = 0.3, aes(color = "y_rep")) +
```

```
  geom_density(data = ds %>% mutate(sim = 1), size=1,
               aes(x = log(birthweight), col = "y")) +
  scale_color_manual(name = "", values = c("y" = "black",
                     "y_rep" = "azure3")) +
  ggtitle("distribution of observed versus predicted birthweights") + theme(plot.title = e
```



Now replot using `ppc_dens_overlay`

```
# Using ppc_dens_overlay for model 2
samp100 <- sample(nrow(yrep2), 100)
ppc_dens_overlay(y, yrep2[samp100, ]) + ggtitle("distribution of observed versus predicted
```

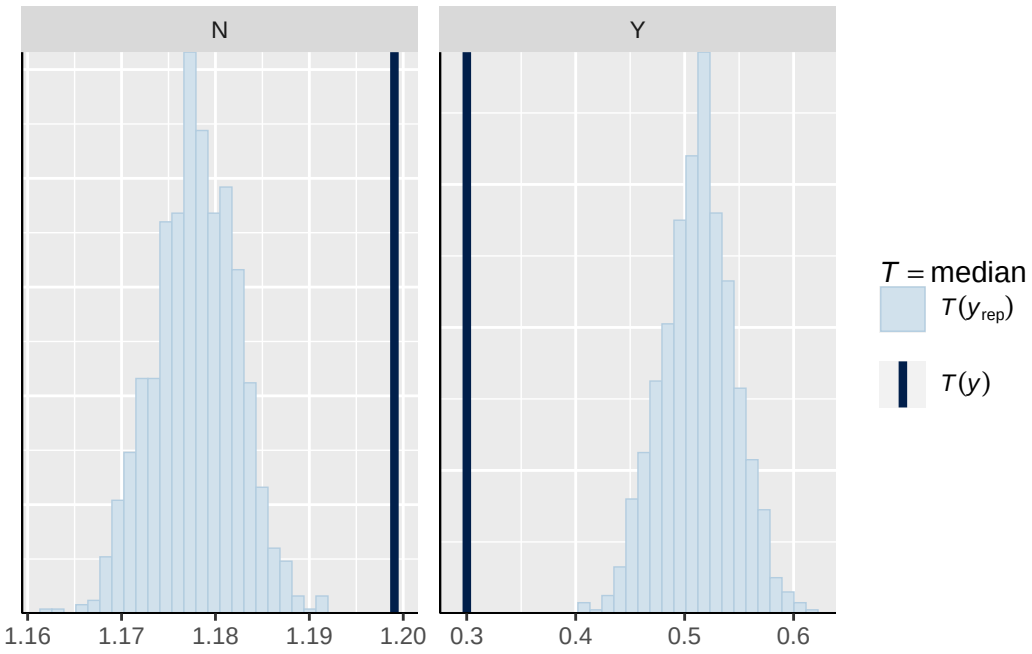


Test statistics

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')
```

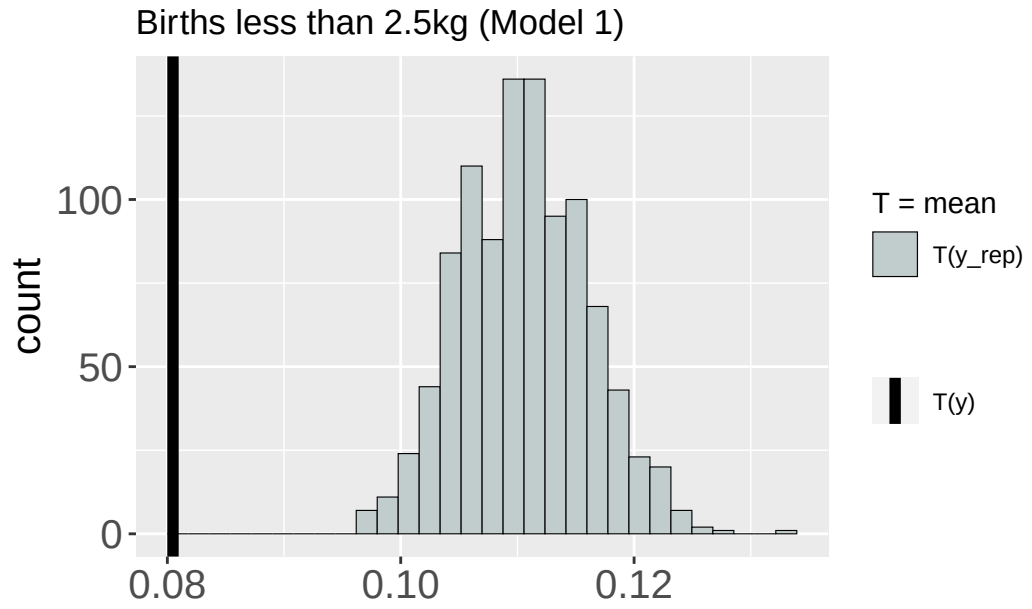


Question 7

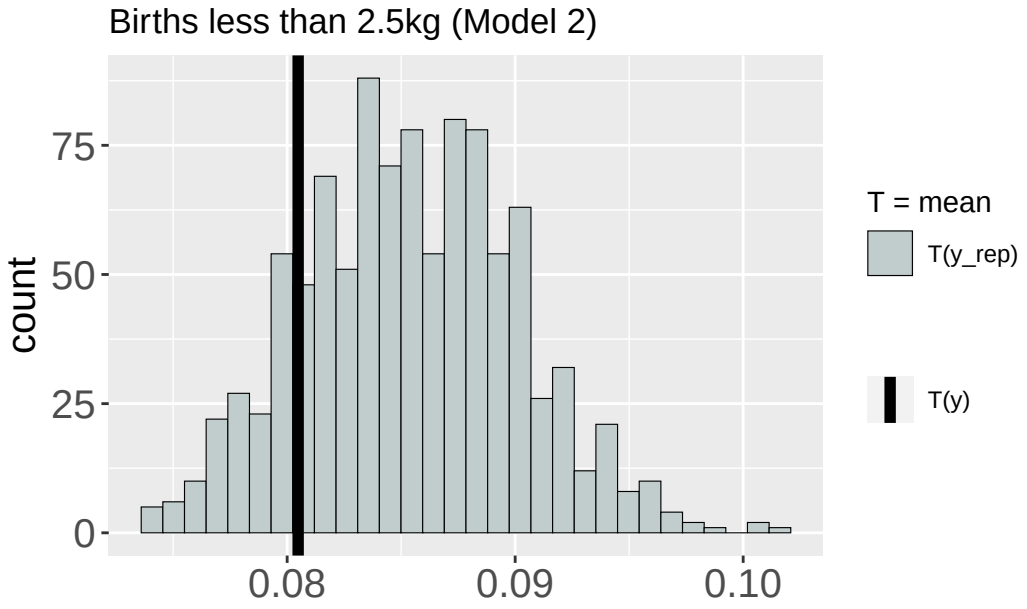
Use a test statistic of the proportion of births under 2.5kg. Calculate the test statistic for the data, and the posterior predictive samples for both models, and plot the comparison (one plot per model).

```
par(mfrow = c(1, 2), mai = c(0.6, 0.6, 0.2, 0.2))
thresh = log(2.5)
#filterd <- function(data, ind, th) {
#  filtered <- mean(data[ind,] <= th)
#  return(filtered)
#}
postpred_1 <- sapply(1:nrow(yrep1),
                     function(i) mean(yrep1[i,] <= thresh))
postpred_2 <- sapply(1:nrow(yrep2),
                     function(i) mean(yrep2[i,] <= thresh))
ythresh <- mean(y <= thresh)

ggplot(data = as_tibble(postpred_1), aes(value)) +
    geom_histogram(col="black", size=0.2, aes(fill = "T(y_rep)")) + labs(x="") + geom_vlin
    color = "T(y)"), lwd=2) +
  ggtitle("Births less than 2.5kg (Model 1)") +
```

```
  scale_color_manual(name = "", values = c("T(y)" = "black")) +
  scale_fill_manual(name = "T = mean",
        values = c("T(y_rep)" = "azure3")) + theme_large_text
```




```
ggplot(data = as_tibble(postpred_2), aes(value)) +
    geom_histogram(col="black", size=0.2, aes(fill = "T(y_rep)")) + labs(x="") + geom_vlin
    color = "T(y)"), lwd=2) +
  ggtitle("Births less than 2.5kg (Model 2)") +
  scale_color_manual(name = "", values = c("T(y)" = "black"))+
  scale_fill_manual(name = "T = mean",
        values = c("T(y_rep)" = "azure3")) + theme_large_text
```



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 3740 log-likelihood matrix

         Estimate    SE
```

```
elpd_loo   1347.9  72.1
p_loo        10.1   1.5
looic     -2695.9 144.2
------
Monte Carlo SE of elpd_loo is 0.1.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     3738  99.9%   689
 (0.5, 0.7]   (ok)          2   0.1%   323
   (0.7, 1]   (bad)         0   0.0%   <NA>
   (1, Inf)   (very bad)    0   0.0%   <NA>

All Pareto k estimates are ok (k < 0.7).
See help('pareto-k-diagnostic') for details.
```

```r
loo2
```

```
Computed from 1000 by 3740 log-likelihood matrix

         Estimate    SE
elpd_loo   1517.7  68.7
p_loo        15.6   2.5
looic     -3035.3 137.4
------
Monte Carlo SE of elpd_loo is 0.2.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     3737  99.9%   425
 (0.5, 0.7]   (ok)          3   0.1%   167
   (0.7, 1]   (bad)         0   0.0%   <NA>
   (1, Inf)   (very bad)    0   0.0%   <NA>

All Pareto k estimates are ok (k < 0.7).
See help('pareto-k-diagnostic') for details.
```

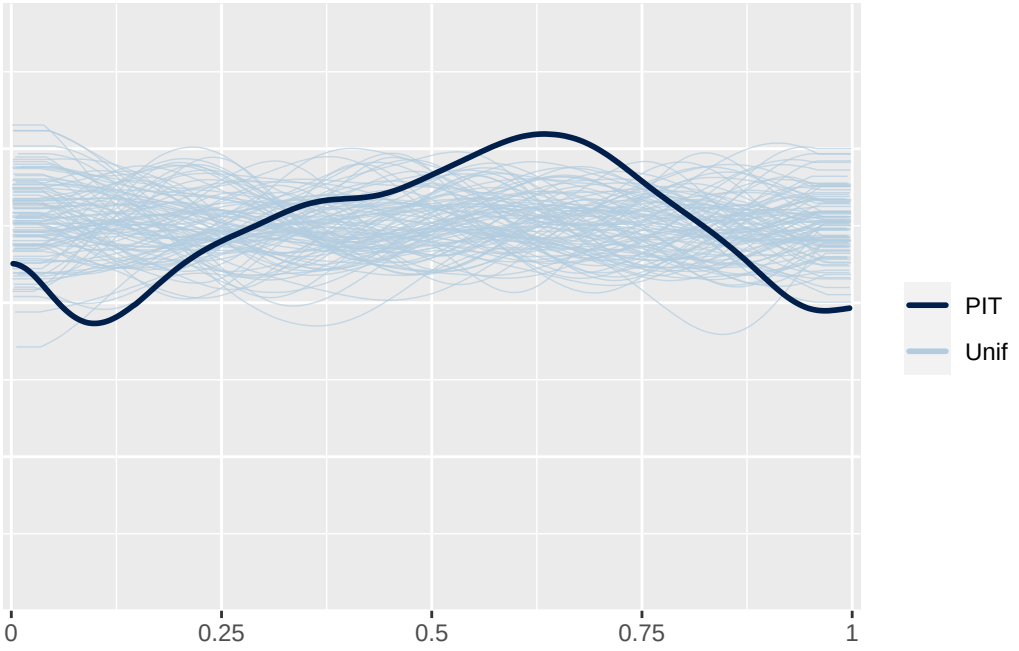
Comparing the two models tells us model 2 is better:

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

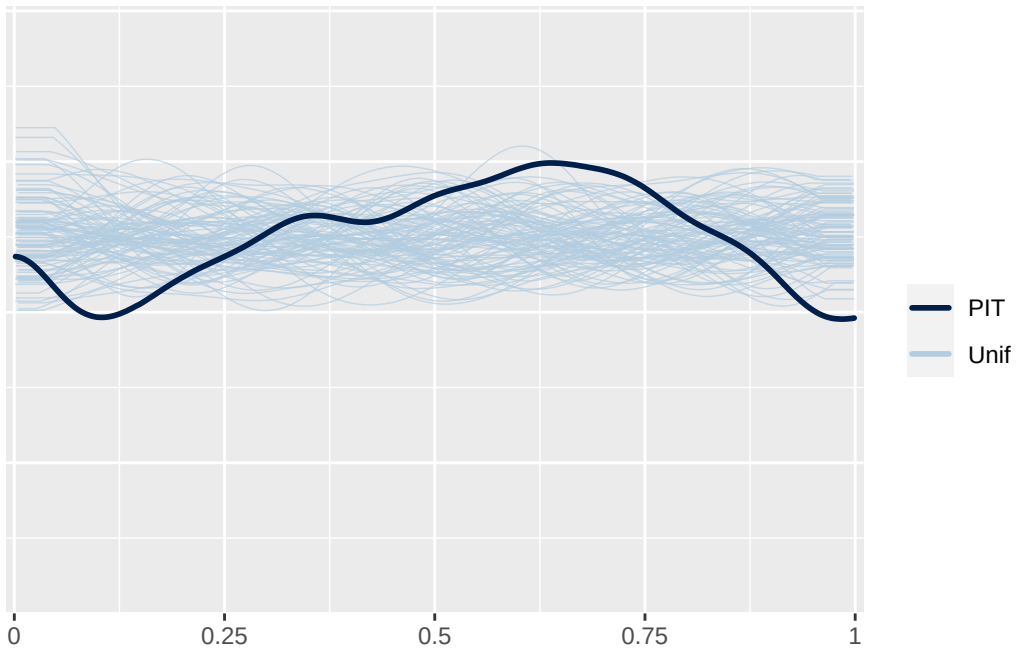
```
       elpd_diff se_diff
model2    0.0       0.0
model1 -169.7      35.8
```

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.

The extra model adds the mother's age (standardized) as a covariate.

```
ds_extra <- ds %>%
  mutate(z = ifelse(ds$preterm=="Y", 1, 0),
         log_age_c = (log(mager) - mean(log(mager)))/sd(log(mager)))
```

Prepare data for Stan

```
stan_data_extra <- list(N = nrow(ds_extra),
                  log_weight = ds_extra$log_weight,
                  log_gest = ds_extra$log_gest_c,
                  z = ds_extra$z,
```

```
                  log_age = ds_extra$log_age_c)
```

```
# Fitting the model
mod_extra <- stan(data = stan_data_extra,
             file = here("./Lab6/mod_extra.stan"),
             iter = 500,
             seed = 243)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.001917 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 19.17 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 1: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 1: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 1: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 1: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 1: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 1: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 1: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 1: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 1: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 1: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 1: Iteration: 500 / 500 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 3.823 seconds (Warm-up)
Chain 1:                5.007 seconds (Sampling)
Chain 1:                8.83 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.001297 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 12.97 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: 7.203 seconds (Warm-up)
Chain 2:                4.569 seconds (Sampling)
Chain 2:                11.772 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.001662 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 16.62 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: 5.13 seconds (Warm-up)
Chain 3:                4.888 seconds (Sampling)
Chain 3:                10.018 seconds (Total)
Chain 3:
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.001068 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 10.68 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: 4.649 seconds (Warm-up)
Chain 4:                3.622 seconds (Sampling)
Chain 4:                8.271 seconds (Total)
Chain 4:
```

Print out the model summary

```
summary(mod_extra)$summary[c(paste0("beta[", 1:5, "]"), "sigma"),]
```

```
              mean      se_mean          sd       2.5%        25%        50%
beta[1] 1.17032183 7.081726e-05 0.002731754 1.16468233 1.16847405 1.17038055
beta[2] 0.10112265 1.161759e-04 0.003574049 0.09415775 0.09870281 0.10098026
beta[3] 0.53794332 3.230268e-03 0.057186511 0.42876356 0.49967211 0.53499840
beta[4] 0.19246123 6.722698e-04 0.012039167 0.17002137 0.18463774 0.19222141
beta[5] 0.01490325 7.281967e-05 0.002577868 0.01013351 0.01313461 0.01483137
sigma   0.16016426 7.803841e-05 0.001840080 0.15669932 0.15886850 0.16021193
               75%      97.5%     n_eff      Rhat
beta[1] 1.17224271 1.17534633 1488.0063 0.9999703
beta[2] 0.10352432 0.10794665  946.4305 1.0031358
beta[3] 0.57257467 0.65727929  313.4081 1.0004775
beta[4] 0.20030154 0.21697202  320.7050 1.0048330
beta[5] 0.01669914 0.01961851 1253.2102 0.9993386
```

```
sigma   0.16139429 0.16377157  555.9767 1.0078410
```

Compare model 2 and model_extra using ELPD computed with leave-one-out cross-validation (LOO-CV)

```
loglik2 <- extract(mod2)[["log_lik"]]
loglik_ext <- extract(mod_extra)[["log_lik"]]
loo <- loo(loglik2, save_psis = TRUE)
loo_ext <- loo(loglik_ext, save_psis = TRUE)
loo2
```

```
Computed from 1000 by 3740 log-likelihood matrix

         Estimate    SE
elpd_loo   1517.7  68.7
p_loo        15.6   2.5
looic     -3035.3 137.4
------
Monte Carlo SE of elpd_loo is 0.2.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     3737  99.9%   425
 (0.5, 0.7]   (ok)          3   0.1%   167
   (0.7, 1]   (bad)         0   0.0%   <NA>
   (1, Inf)   (very bad)    0   0.0%   <NA>

All Pareto k estimates are ok (k < 0.7).
See help('pareto-k-diagnostic') for details.
```

```
loo_ext
```

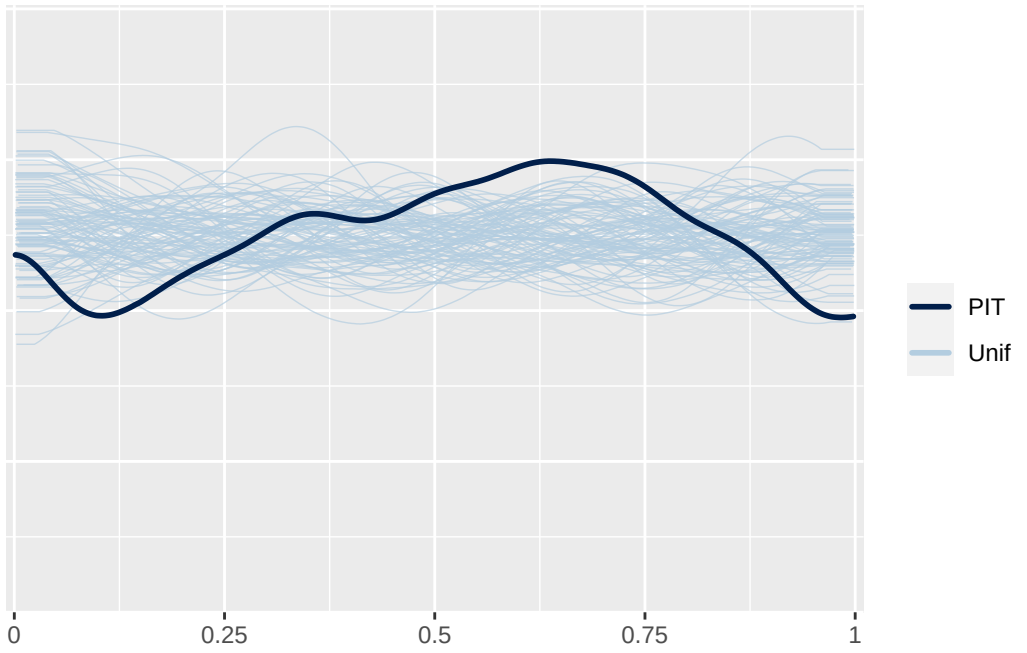
```
Computed from 1000 by 3740 log-likelihood matrix

         Estimate    SE
elpd_loo   1533.7  69.4
p_loo        15.5   2.3
looic     -3067.3 138.9
```

```
------
Monte Carlo SE of elpd_loo is NA.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     3738  99.9%   347
 (0.5, 0.7]   (ok)          1   0.0%   354
   (0.7, 1]   (bad)         1   0.0%   232
   (1, Inf)   (very bad)    0   0.0%   <NA>
See help('pareto-k-diagnostic') for details.

       elpd_diff se_diff
model2   0.0       0.0
model1 -16.0       5.6
```

Compare model 2 and model_extra using LOO-PIT

```
yrep2 <- extract(mod2)[["log_weight_rep"]]
yrep_ext <- extract(mod_extra)[["log_weight_rep"]]
```

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



```
ppc_loo_pit_overlay(yrep = yrep_ext, y = y, lw = weights(loo_ext$psis_object))
```