Lab Exercises 6

19/02/24

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("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

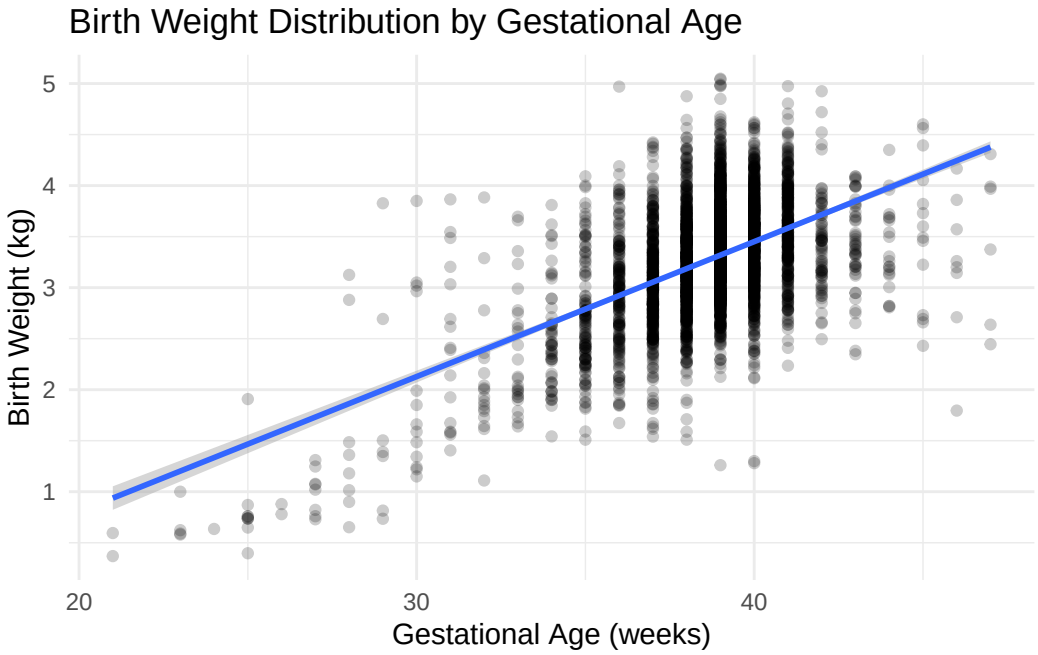
Use plots or tables to show three interesting observations about the data. Remember:

- 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.

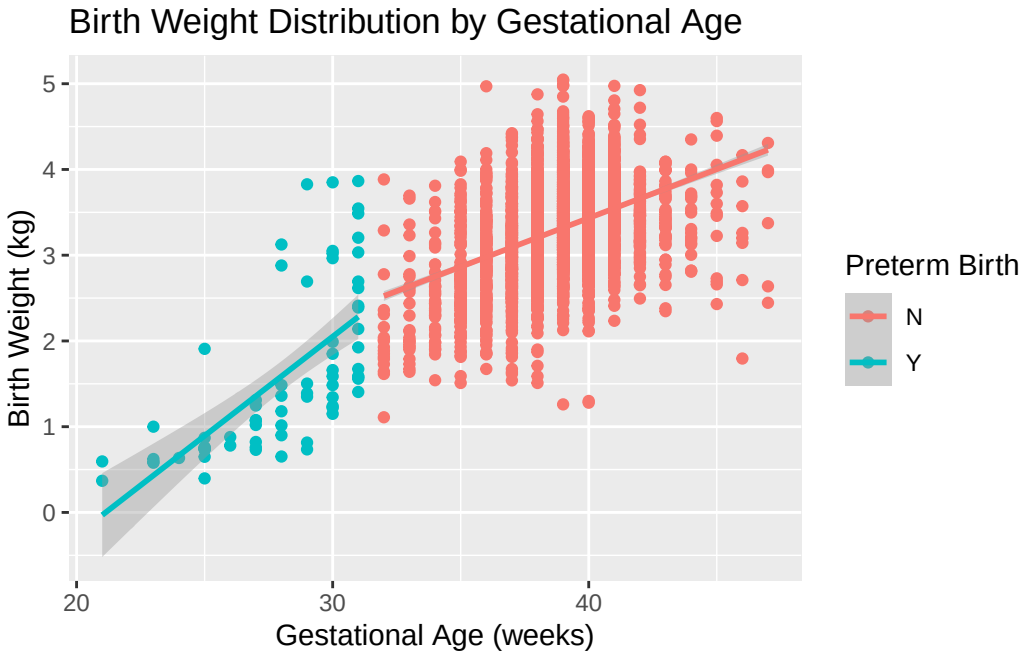
```
ggplot(ds, aes(x = gest, y = birthweight)) +
  geom_point(alpha = 0.2) +
  geom_smooth(method = "lm") +
  labs(title = "Birth Weight Distribution by Gestational Age",
       x = "Gestational Age (weeks)",
```

```
      y = "Birth Weight (kg)") +
  theme_minimal()
```



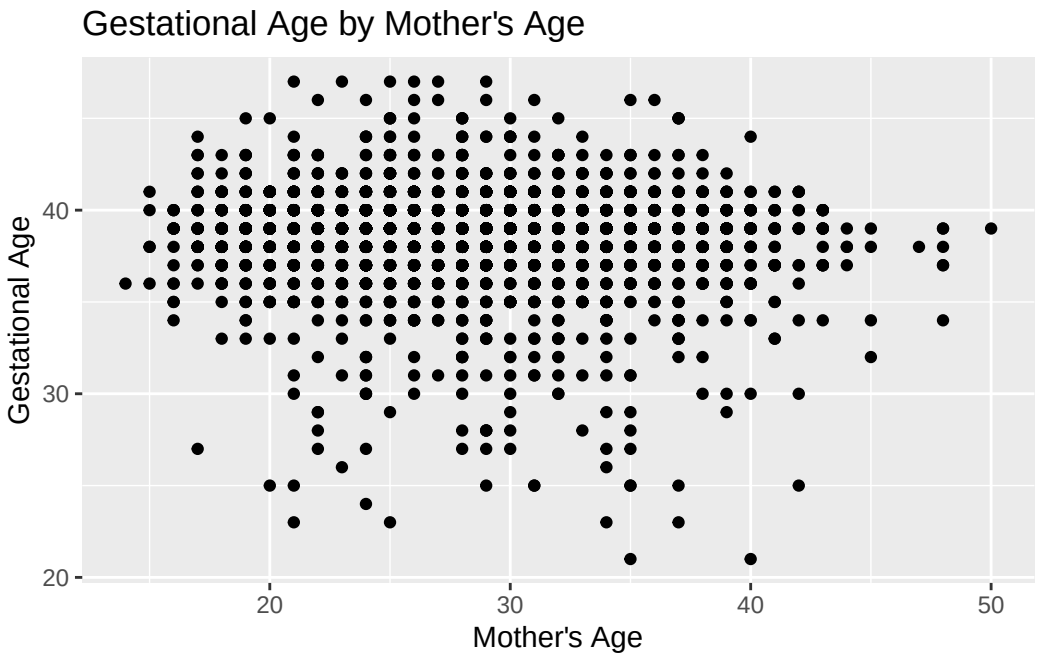
As the gestational age increases, birth weight of the baby increases. They have a positive relationship.

```
ds |>
  ggplot(aes(x = gest, y = birthweight, color = as.factor(preterm))) +
  geom_point() +
  geom_smooth(method = 'lm', aes(group = preterm)) +
  labs(title = "Birth Weight Distribution by Gestational Age",
       x = "Gestational Age (weeks)",
       y = "Birth Weight (kg)",
       color = "Preterm Birth")
```



Green points represent preterm and red points represent normal term. You can see that the relationship between the birth weight and gestational age for preterm babies looks more sensitive than the normal term babies because they have a sharper slope.

```
ggplot(ds, aes(x = mager, y = gest)) +
geom_point() +
labs(title = "Gestational Age by Mother's Age",
      x = "Mother's Age",
      y = "Gestational Age")
```



There is no clear positive or negative trends between the mom's age and gestational age. The one you can notice is that, for mom's age between 20 and 35, the gestational age is similarly distributed, however, for outside of range of the age, they have a narrower distributions of gestational age.

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

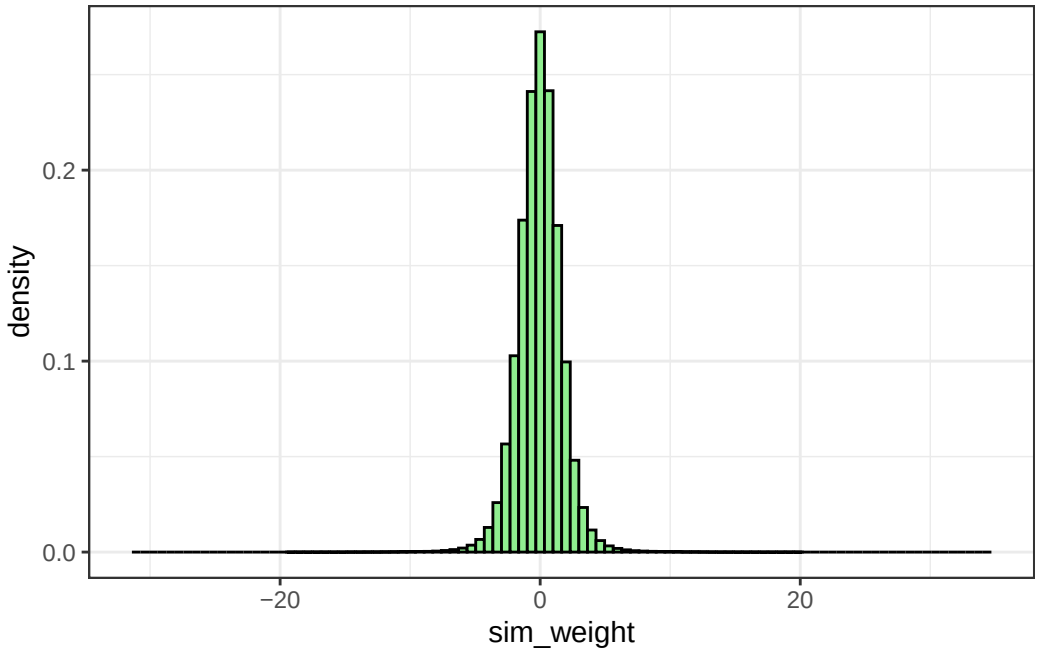
For Model 1, simulate values of βs and σ based on the priors above. Do 1000 simulations. Use these values to simulate (log) birth weights from the likelihood specified in Model 1, based on the set of observed gestational weights. **Remember the gestational weights should be centered and standardized**.

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

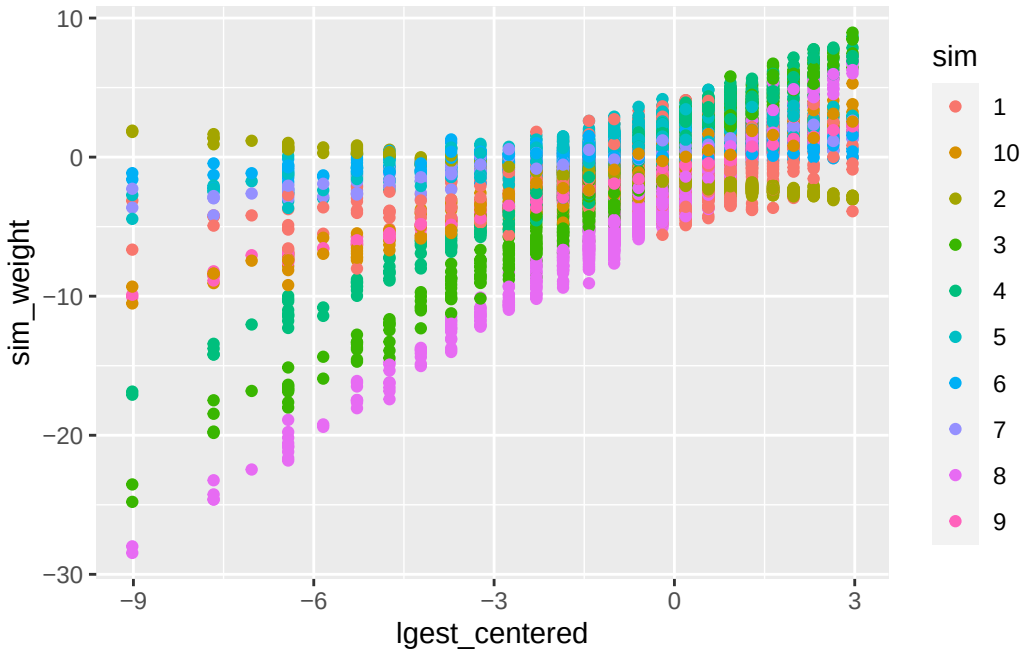
```
n_sim <- 1000
sigma <- abs(rnorm(n_sim, 0, 1))
beta0 <- rnorm(n_sim, 0, 1)
beta1 <- rnorm(n_sim, 0, 1)

simulation <- tibble(lgest_centered = (log(ds$gest) - mean(log(ds$gest)))/sd(log(ds$gest))

for(i in 1:n_sim){
  mu <- beta0[i] + beta1[i]*simulation$lgest_centered
  simulation[paste0(i)] <- mu + rnorm(nrow(simulation), 0, sigma[i])
}
```

```
ds_sim <- simulation |>
  pivot_longer(`1`:`1000`, names_to = "sim", values_to = "sim_weight")
```

```
ds_sim %>%
  ggplot(aes(sim_weight)) + geom_histogram(aes(y = ..density..), bins = 100, fill = "light
  theme_bw()
```




```
simulation[, 1:11] |>
  pivot_longer(`1`:`10`, names_to = "sim", values_to = "sim_weight") |>
  ggplot(aes(x = lgest_centered, y = sim_weight, color = sim)) +
  geom_point()
```



Run the model

Now we're going to run Model 1 in Stan. The stan code is in the `code/models` folder.

First, get our data into right form for input into stan.

```
ds$log_weight <- log(ds$birthweight)
ds$log_gest_c <- (log(ds$gest) - mean(log(ds$gest)))/sd(log(ds$gest))

# put into a list
stan_data <- list(N = nrow(ds),
                  log_weight = ds$log_weight,
                  log_gest = ds$log_gest_c)
```

Now fit the model

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

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000353 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 3.53 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: 0.37 seconds (Warm-up)
Chain 1:                0.314 seconds (Sampling)
Chain 1:                0.684 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000118 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.18 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: 0.341 seconds (Warm-up)
Chain 2:                0.293 seconds (Sampling)
Chain 2:                0.634 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000146 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.46 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: 0.356 seconds (Warm-up)
Chain 3:                0.294 seconds (Sampling)
Chain 3:                0.65 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000213 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.13 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: 0.364 seconds (Warm-up)
Chain 4:                0.285 seconds (Sampling)
Chain 4:                0.649 seconds (Total)
Chain 4:
```

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

```
            mean      se_mean          sd      2.5%       25%       50%
beta[1] 1.1625576 7.819055e-05 0.002741220 1.1569041 1.1608022 1.1626034
beta[2] 0.1437597 6.826883e-05 0.002727365 0.1381289 0.1418676 0.1437582
sigma   0.1689121 1.079222e-04 0.001918795 0.1651348 0.1677272 0.1689781
             75%     97.5%     n_eff      Rhat
beta[1] 1.1644381 1.1676158 1229.0775 0.9967375
beta[2] 0.1456099 0.1490155 1596.0319 0.9969380
sigma   0.1701247 0.1724931  316.1081 1.0038777
```

Question 3

Based on Model 1, give an estimate of the expected birthweight of a baby who was born at a gestational age of 37 weeks.

```
adj_gest <- (log(37) - mean(log(ds$gest)))/sd(log(ds$gest))
samples <- extract(mod1)
```

```
median(exp(samples$beta[,1] + adj_gest*samples$beta[,2]))
```

```
[1] 2.936136
```

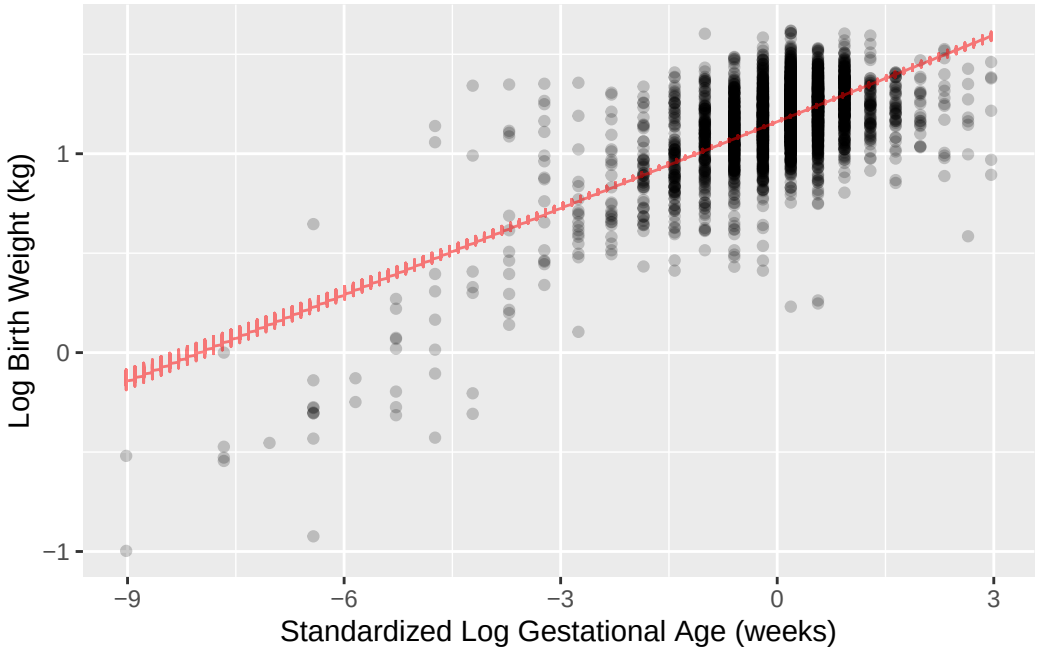
Question 4

Based on Model 1, create a scatter plot showing the underlying data (on the appropriate scale) and 50 posterior draws of the linear predictor.

```
post_draws <- as.data.frame(extract(mod1))
set.seed(1010138067) # Set a seed for reproducibility
draws_indices <- sample(1:nrow(post_draws), 50)
draws <- post_draws[draws_indices, ]

# Generate a sequence for gestational age to plot the regression lines
gest_seq <- seq(from = min(ds$log_gest_c), to = max(ds$log_gest_c), length.out = 100)
```

```
regression_lines <- expand.grid(
  log_gest_c = gest_seq,
  draw = draws_indices) |>
  mutate(
    beta1 = post_draws$beta.1[draw],
    beta2 = post_draws$beta.2[draw],
    lweight_pred = beta1 + beta2 * log_gest_c
  )
```

```
ggplot(ds, aes(x = log_gest_c, y = log_weight)) +
  geom_point(alpha = 0.2) +
  labs(x = "Standardized Log Gestational Age (weeks)", y = "Log Birth Weight (kg)") +
  geom_line(data = regression_lines, aes(y = lweight_pred), color = "red", alpha = 0.5) +
  theme(legend.position = "none")
```



Question 5

Write a Stan model to run Model 2, and run it. Report a summary of the results, and interpret the coefficient estimate on the interaction term.

```
ds$log_weight <- log(ds$birthweight)
ds$log_gest_c <- (log(ds$gest) - mean(log(ds$gest))) / sd(log(ds$gest))
ds$preterm <- ifelse(ds$preterm == "Y", 1, 0)

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

```
mod2 <- stan(data = stan_data,
             file = "simple_weight2.stan",
             iter = 500,
             seed = 243)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000536 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 5.36 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 1: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 1: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 1: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 1: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 1: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 1: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 1: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 1: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 1: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 1: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 1: Iteration: 500 / 500 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 1.16 seconds (Warm-up)
Chain 1:                0.883 seconds (Sampling)
Chain 1:                2.043 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000206 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.06 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 2: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 2: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 2: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 2: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 2: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 2: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 2: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 2: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 2: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 2: Iteration: 450 / 500 [ 90%]  (Sampling)
```

```
Chain 2: Iteration: 500 / 500 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 1.12 seconds (Warm-up)
Chain 2:                1.091 seconds (Sampling)
Chain 2:                2.211 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.00026 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.6 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 3: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 3: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 3: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 3: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 3: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 3: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 3: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 3: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 3: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 3: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 3: Iteration: 500 / 500 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 1.177 seconds (Warm-up)
Chain 3:                0.914 seconds (Sampling)
Chain 3:                2.091 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000242 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.42 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 4: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 4: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 4: Iteration: 150 / 500 [ 30%]  (Warmup)
```

```
Chain 4: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 4: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 4: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 4: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 4: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 4: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 4: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 4: Iteration: 500 / 500 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 1.148 seconds (Warm-up)
Chain 4:                0.874 seconds (Sampling)
Chain 4:                2.022 seconds (Total)
Chain 4:
```

```r
summary(mod2)$summary["beta[4]",]
```

```
       mean      se_mean           sd         2.5%          25%          50%
1.975011e-01 6.522811e-04 1.267048e-02 1.745808e-01 1.885280e-01 1.970499e-01
         75%        97.5%        n_eff         Rhat
2.066981e-01 2.233768e-01 3.773260e+02 1.003674e+00
```

From the result, you can know that when both preterm and log_gest_c increase by one unit, the log_weight is expected to increase by an average of 0.1975 units, holding all other variables constant.

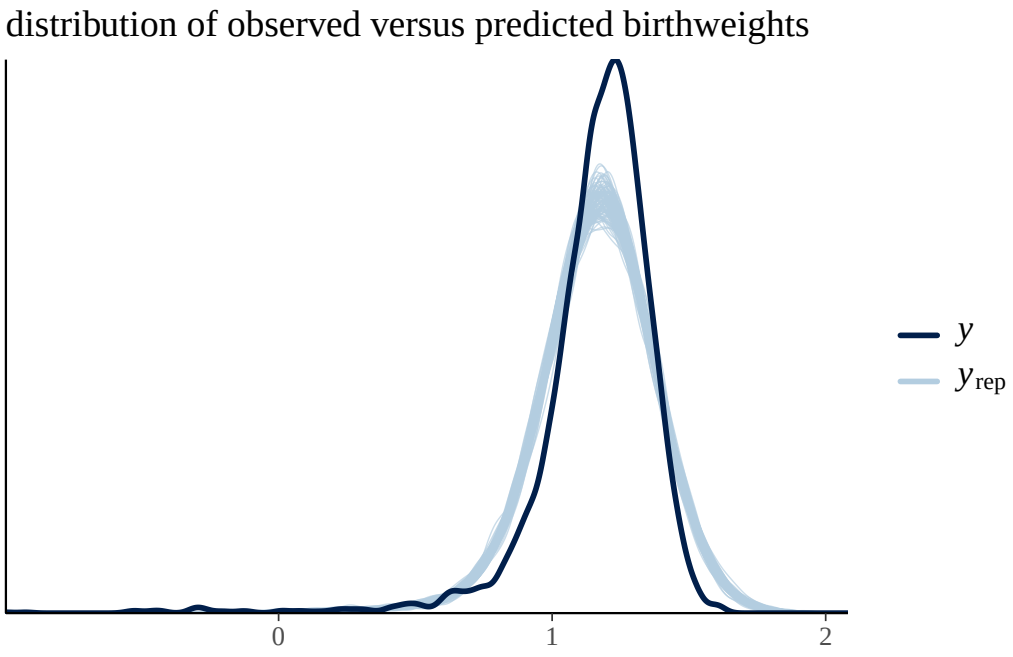
PPCs

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:

```r
set.seed(1856)
y <- ds$log_weight
yrep1 <- extract(mod1)[["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

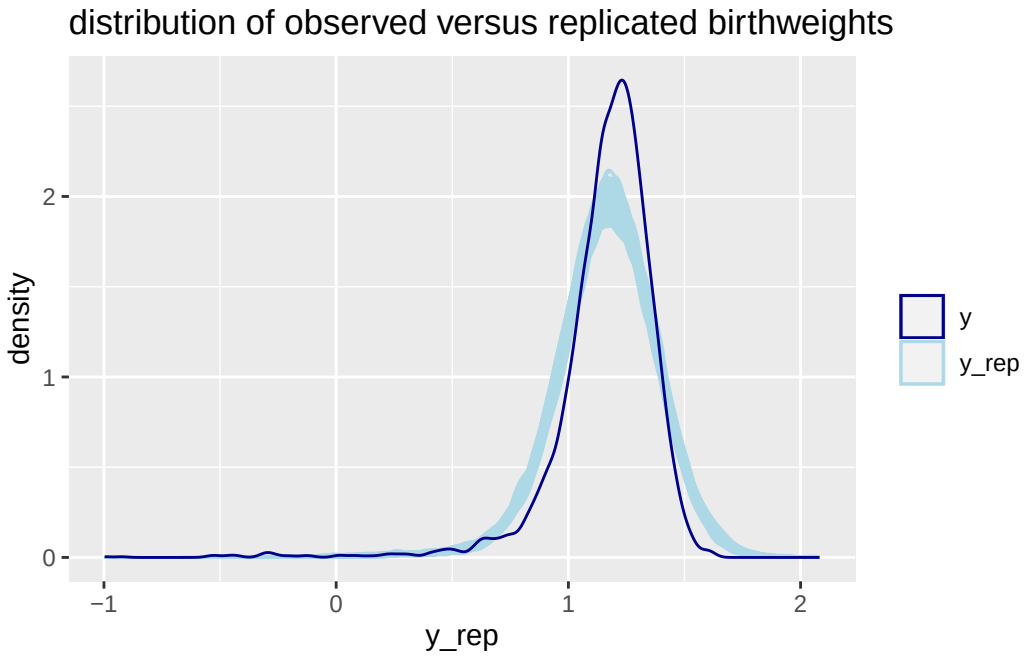
Make a similar plot to the one above but for Model 2, and **not** using the bayes plot in built function (i.e. do it yourself just with `geom_density`)

```
rownames(yrep1) <- 1:nrow(yrep1)
drep <- as_tibble(t(yrep1))
drep <- drep |>
  bind_cols(i = 1:nrow(ds), log_weight_obs = log(ds$birthweight))
```

```
drep <- drep |>
  pivot_longer(-(i:log_weight_obs), names_to = "sim", values_to ="y_rep")
```

```
drep |>
  filter(sim %in% samp100) |>
  ggplot(aes(y_rep, group = sim)) +
```

```
geom_density(alpha = 0.2, aes(color = "y_rep")) +
geom_density(data = ds |> mutate(sim = 1),
             aes(x = log(birthweight), col = "y")) +
scale_color_manual(name = "",
                   values = c("y" = "darkblue", "y_rep" = "lightblue")) +
ggtitle("distribution of observed versus replicated birthweights")
```

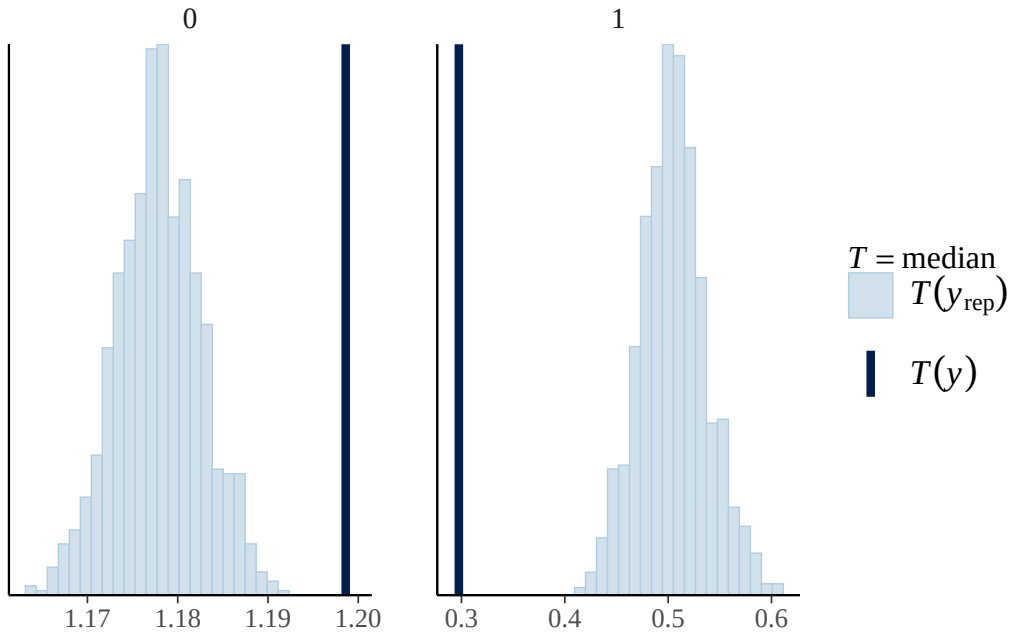


Test statistics

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')
```

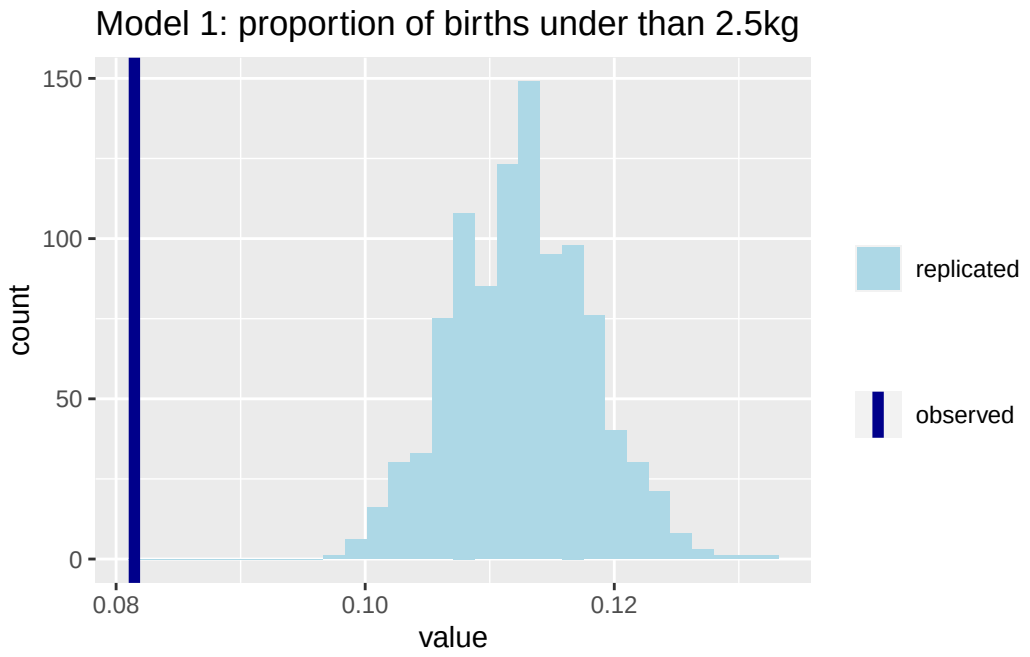


Question 7

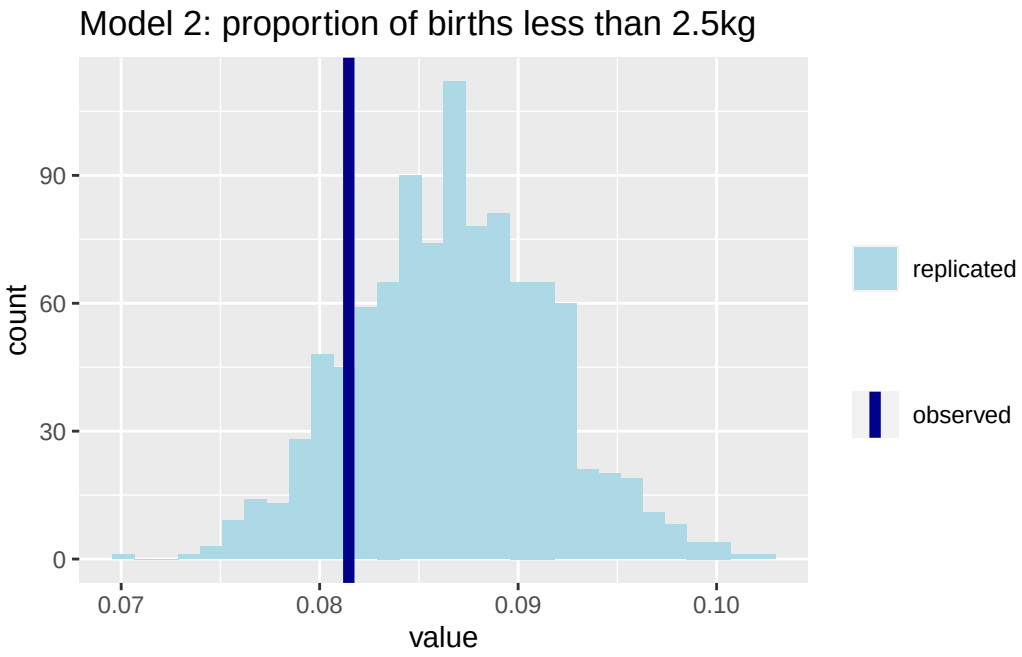
Use a test statistic of the proportion of births under 2.5kg. Calculate the test statistic for the data, and the posterior predictive samples for both models, and plot the comparison (one plot per model).

```
yrep2 <- extract(mod2)[["log_weight_rep"]]
test_y <- mean(y < log(2.5))
test_y_rep <- sapply(1:nrow(yrep1), function(i) mean(yrep1[i,] < log(2.5)))
test_y_rep2 <- sapply(1:nrow(yrep2), function(i) mean(yrep2[i,] < log(2.5)))
```

```
ggplot(data = as_tibble(test_y_rep), aes(value)) +
    geom_histogram(aes(fill = "replicated")) +
    geom_vline(aes(xintercept = test_y, color = "observed"), lwd = 2.0) +
  ggtitle("Model 1: proportion of births under than 2.5kg") +
  scale_color_manual(name = "",
                     values = c("observed" = "darkblue"))+
  scale_fill_manual(name = "",
                     values = c("replicated" = "lightblue"))
```




```
ggplot(data = as_tibble(test_y_rep2), aes(value)) +
    geom_histogram(aes(fill = "replicated")) +
    geom_vline(aes(xintercept = test_y, color = "observed"), lwd = 2.0) +
  ggtitle("Model 2: proportion of births less than 2.5kg") +
  scale_color_manual(name = "",
                     values = c("observed" = "darkblue"))+
  scale_fill_manual(name = "",
                     values = c("replicated" = "lightblue"))
```



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"]]
```

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)
```

Look at the output:

```
loo1
```

```
Computed from 1000 by 3842 log-likelihood matrix

         Estimate    SE
```

```
elpd_loo   1377.4  72.5
p_loo         9.3   1.3
looic     -2754.8 145.0
------
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.
```

Question 8

Get the LOO estimate of elpd for Model 2 and compare the two models with the `loo_compare` function. Interpret the results.

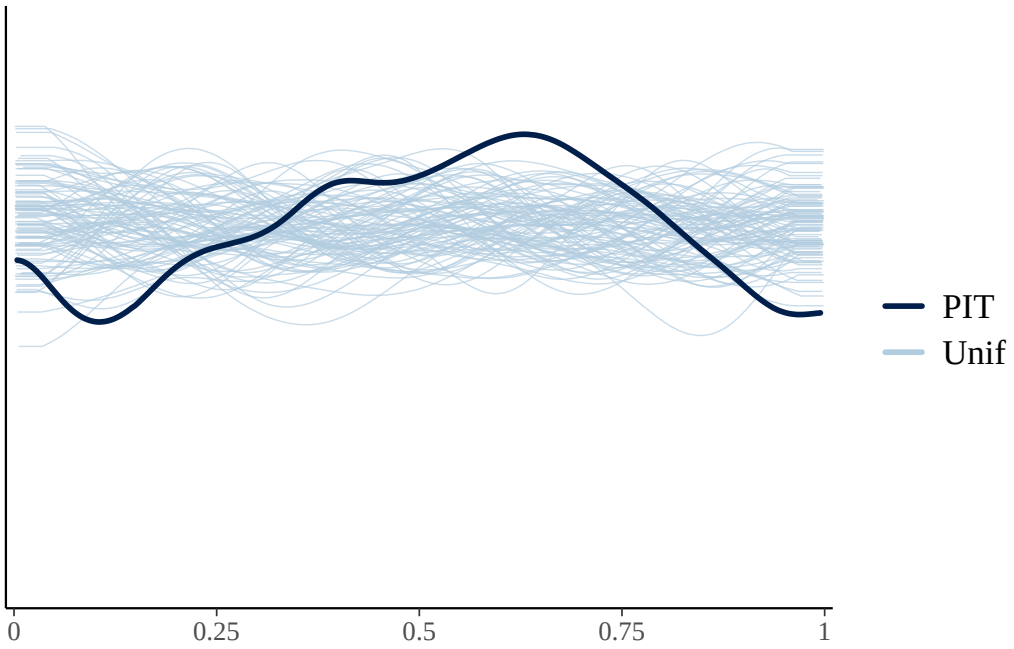
```r
loglik_mod2 <- extract(mod2)[["log_lik"]]
loo2 <- loo(loglik_mod2, save_psis = TRUE)
loo_compare(loo1, loo2)
```

```
       elpd_diff se_diff
model2    0.0       0.0
model1 -176.1      36.3
```

Model2 has a higher elpd, which means a better prediction performance.

We can also compare the LOO-PIT of each of the models to standard uniforms. For example for Model 1:

```r
library(rstantools)
ppc_loo_pit_overlay(yrep = yrep1, y = y, lw = weights(loo1$psis_object))
```



Bonus question (not required)

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

Question 9

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.