App stat 2 lab_exercise6.qmd

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.2     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```
library(here)
```

```
here() starts at /Users/euijinbaek/STA2201
```

```
# for bayes stuff
library(rstan)
```

```
Loading required package: StanHeaders

rstan version 2.32.5 (Stan version 2.32.2)

For execution on a local, multicore CPU with excess RAM we recommend calling
options(mc.cores = parallel::detectCores()).
To avoid recompilation of unchanged Stan programs, we recommend calling
rstan_options(auto_write = TRUE)
```

```
For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
change `threads_per_chain` option:
rstan_options(threads_per_chain = 1)


Attaching package: 'rstan'

The following object is masked from 'package:tidyr':

    extract
```

```
library(bayesplot)
```

```
This is bayesplot version 1.11.0
- Online documentation and vignettes at mc-stan.org/bayesplot
- bayesplot theme set to bayesplot::theme_default()
   * Does _not_ affect other ggplot2 plots
   * See ?bayesplot_theme_set for details on theme setting
```

```
library(loo)
```

```
This is loo version 2.6.0
- Online documentation and vignettes at mc-stan.org/loo
- As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'co

Attaching package: 'loo'

The following object is masked from 'package:rstan':

    loo
```

```
library(tidybayes)

ds <- read_rds("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
```

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)
head(ds)
```

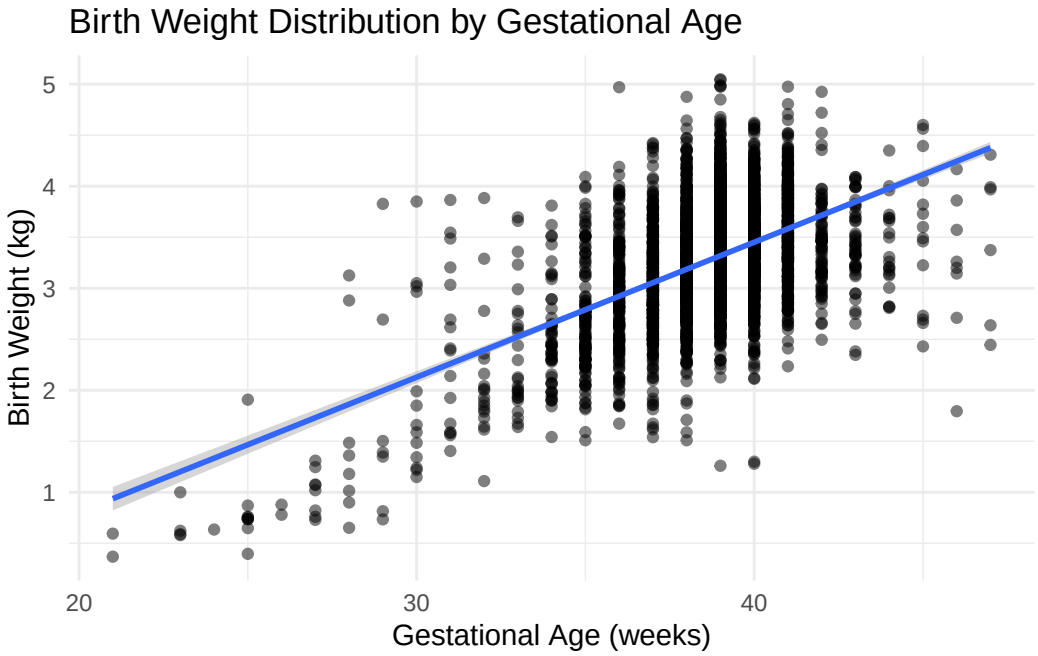
```
# A tibble: 6 x 9
  mager mracehisp meduc   bmi sex    gest birthweight ilive preterm
  <dbl>     <dbl> <dbl> <dbl> <chr> <dbl>       <dbl> <chr> <chr>
1    16         2     2  23   M        39        3.18 Y     N
2    25         7     2  43.6 M        40        4.14 Y     N
3    27         2   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

1

It seems that there is positive correlation between gestational age and birth weight.

```
ggplot(ds, aes(x = gest, y = birthweight)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm") +
  labs(title = "Birth Weight Distribution by Gestational Age",
       x = "Gestational Age (weeks)",
       y = "Birth Weight (kg)") +
  theme_minimal()
```

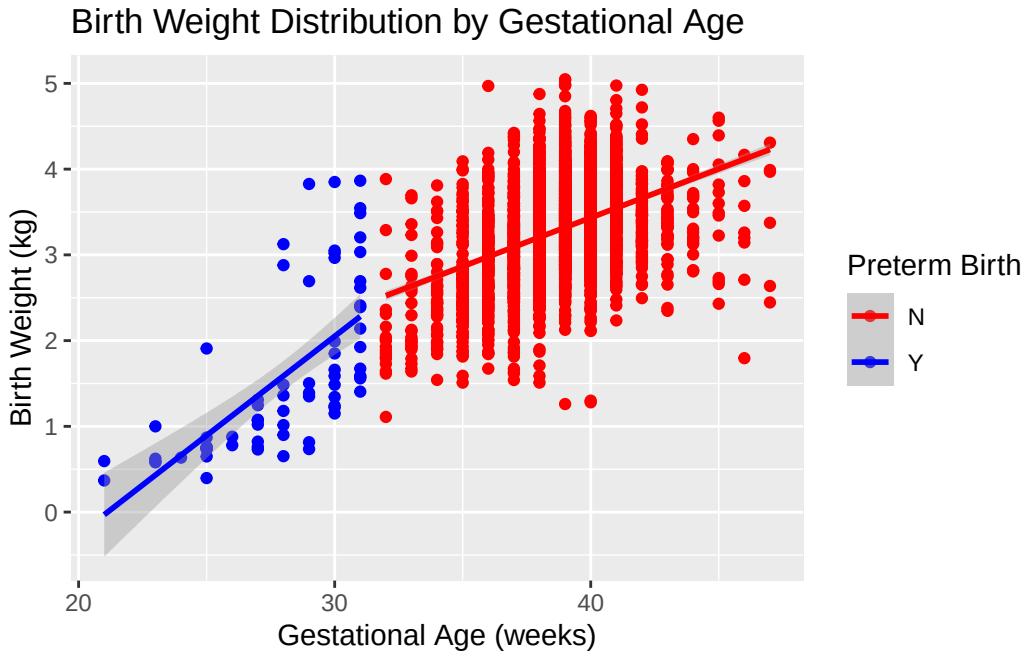
```
`geom_smooth()` using formula = 'y ~ x'
```



The slope of the line for preterm births (blue) appears steeper than that for full-term births (red), suggesting that weekly weight gain may be greater at earlier stages of gestation.

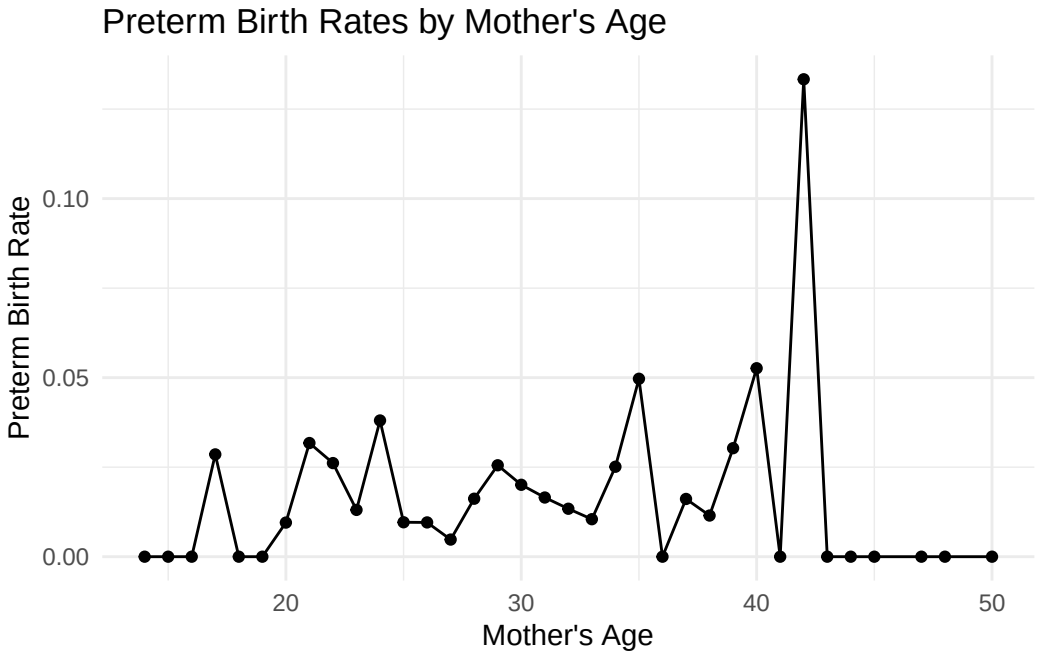
```
ds |>
  ggplot(aes(x = gest, y = birthweight, color = as.factor(preterm))) +
  geom_point() +
  geom_smooth(method = 'lm', aes(group = preterm)) +
  labs(title = "Birth Weight Distribution by Gestational Age",
       x = "Gestational Age (weeks)",
       y = "Birth Weight (kg)",
       color = "Preterm Birth") +
  scale_color_manual(values = c('Y' = 'blue', 'N' = 'red'),
                     labels = c('Preterm' = 'Yes', 'Full-Term' = 'No'))
```

```
`geom_smooth()` using formula = 'y ~ x'
```



There seem to be notably higher rates of preterm births at certain ages, most dramatically in the age range of 40s. This could suggest increased risks associated with advanced maternal age.

```
ds %>%
  group_by(mager) |>
  summarise(preterm_rate = mean(preterm == "Y")) |>
  ggplot(aes(x = mager, y = preterm_rate)) +
  geom_line() +
  geom_point() +
  labs(title = "Preterm Birth Rates by Mother's Age",
       x = "Mother's Age",
       y = "Preterm Birth Rate") +
  theme_minimal()
```



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.

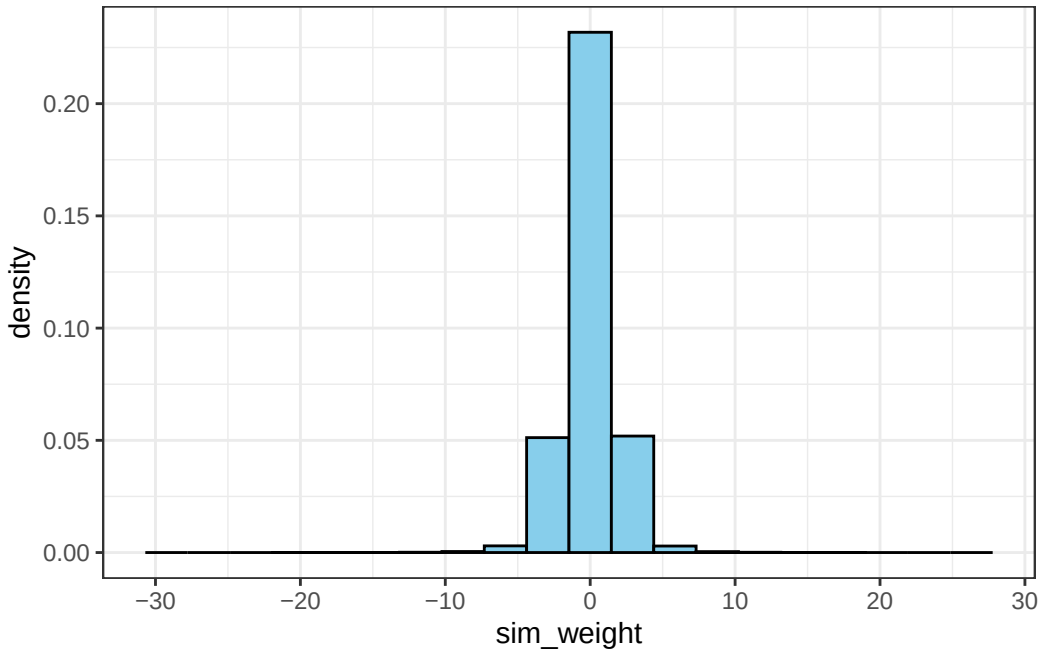
2.

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**.

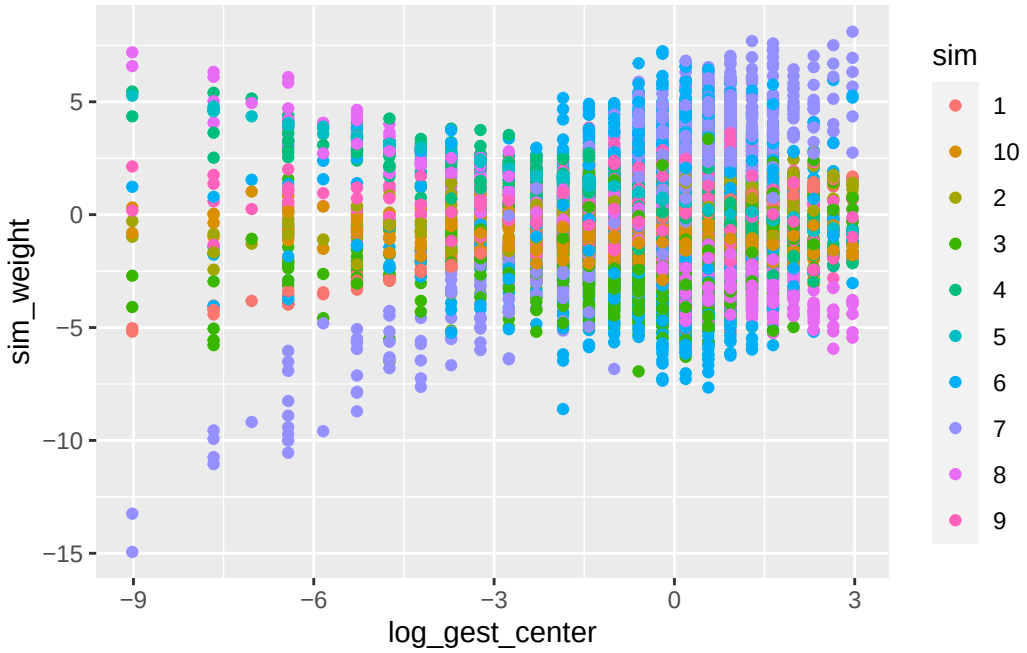
- Plot the resulting distribution of simulated (log) birth weights.
- Plot ten simulations of (log) birthweights against gestational age.

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

sims <- tibble(log_gest_center = (log(ds$gest) - mean(log(ds$gest)))/sd(log(ds$gest)))

for(i in 1:nsims){
  mu <- beta0[i] + beta1[i]*sims$log_gest_center
  sims[paste0(i)] <- mu + rnorm(nrow(sims), 0, sigma[i])
}


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

dsl %>%
  ggplot(aes(sim_weight)) + geom_histogram(aes(y = ..density..), bins = 20, fill = "skyblu
  theme_bw()
```

```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```




```
sims[,1:11] |>
  pivot_longer(`1`:`10`, names_to = "sim", values_to = "sim_weight") |>
  ggplot(aes(x=log_gest_center, 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 = "code/models/simple_weight.stan",
             iter = 500,
             seed = 243)
```

```
Warning in readLines(file, warn = TRUE): incomplete final line found on
'/Users/euijinbaek/STA2201/labs/code/models/simple_weight.stan'
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000143 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.43 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: 0.208 seconds (Warm-up)
Chain 1:                0.164 seconds (Sampling)
Chain 1:                0.372 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 7.1e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.71 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: 0.201 seconds (Warm-up)
Chain 2:                0.193 seconds (Sampling)
Chain 2:                0.394 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 7.6e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.76 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: 0.206 seconds (Warm-up)
Chain 3:                0.179 seconds (Sampling)
Chain 3:                0.385 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 8.4e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.84 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: 0.201 seconds (Warm-up)
Chain 4:                0.179 seconds (Sampling)
Chain 4:                0.38 seconds (Total)
Chain 4:

Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and ta
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#tail-ess
```

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

```
             mean      se_mean          sd      2.5%       25%       50%
beta[1] 1.1626293 8.109795e-05 0.002794886 1.1571530 1.1607401 1.1626100
beta[2] 0.1437074 7.050797e-05 0.002760482 0.1381359 0.1418908 0.1436313
sigma   0.1688448 1.025235e-04 0.001845326 0.1652251 0.1675565 0.1689364
              75%     97.5%    n_eff      Rhat
beta[1] 1.1646213 1.1679552 1187.705 0.9970491
beta[2] 0.1454944 0.1491751 1532.828 0.9972723
sigma   0.1700804 0.1722141  323.966 1.0095667
```

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.

```r
adjusted_gest <- (log(37) - mean(log(ds$gest)))/sd(log(ds$gest))
samples <- extract(mod1)
# Estimated value using median
median(exp(samples[["beta"]][,1] + adjusted_gest*samples[["beta"]][,2]))
```

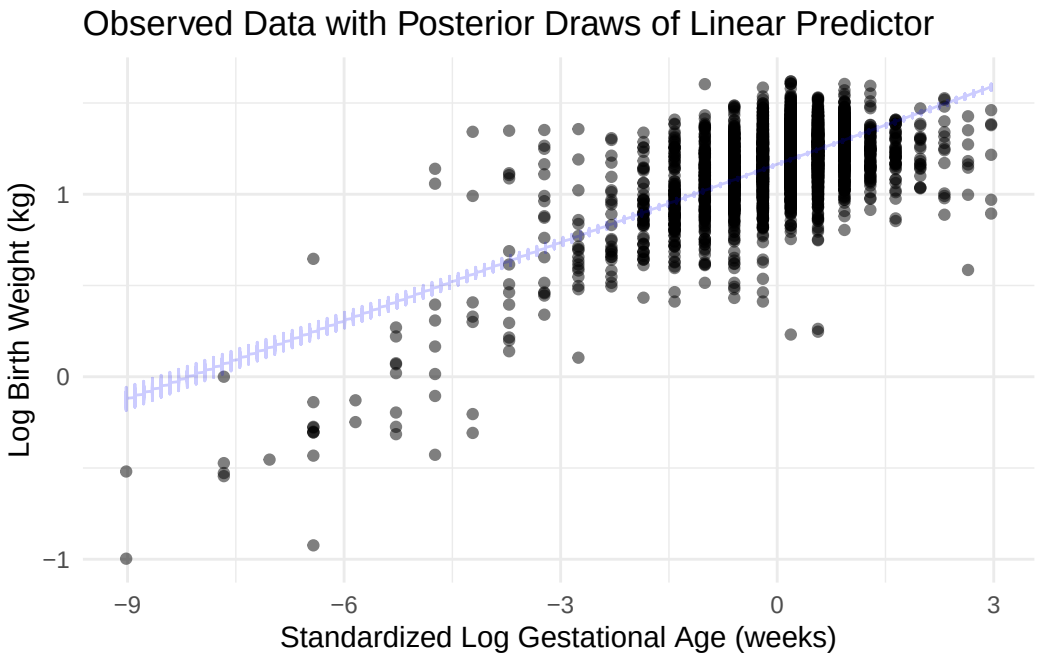
```
[1] 2.936571
```

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.

```r
# Extracting a sample of posterior draws
posterior_draws <- as.data.frame(extract(mod1))

# Selecting 50 random posterior draws for the linear predictor
set.seed(123) # Set a seed for reproducibility
sample_draws_indices <- sample(1:nrow(posterior_draws), 50)
```

```r
# Extracting the specific draws
selected_draws <- posterior_draws[sample_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)

# Creating a data frame for the regression lines
regression_lines <- expand.grid(
  log_gest_c = gest_seq,
  draw = sample_draws_indices
) |>
  mutate(
    beta1 = posterior_draws$beta.1[draw], # Corrected column names
    beta2 = posterior_draws$beta.2[draw], # Corrected column names
    log_weight_pred = beta1 + beta2 * log_gest_c
  )

# Creating the scatter plot with log-transformed variables
ggplot(ds, aes(x = log_gest_c, y = log_weight)) + # Corrected aes placement
  geom_point(alpha = 0.5, color = "black") +
  labs(x = "Standardized Log Gestational Age (weeks)", y = "Log Birth Weight (kg)",
       title = "Observed Data with Posterior Draws of Linear Predictor") +
  theme_minimal() +
  # Adding the 50 posterior draws of the linear predictor
  geom_line(data = regression_lines, aes(y = log_weight_pred), color = "blue", alpha = 0.2
  theme(legend.position = "none")
```



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

stan_data <- list(N = nrow(ds),
                  log_weight = ds$log_weight,
                  log_gest = ds$log_gest_c,
                  prematurity = ds$prematurity,
                  interac = ds$prematurity*ds$log_gest_c)

mod2 <- stan(data = stan_data,
             file = "code/models/simple_weight_mod2.stan",
             iter = 500,
             seed = 243)
```

```
Warning in readLines(file, warn = TRUE): incomplete final line found on
'/Users/euijinbaek/STA2201/labs/code/models/simple_weight_mod2.stan'
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000291 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.91 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: 0.67 seconds (Warm-up)
Chain 1:                0.53 seconds (Sampling)
Chain 1:                1.2 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000151 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.51 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 2: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 2: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 2: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 2: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 2: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 2: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 2: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 2: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 2: Iteration: 400 / 500 [ 80%]  (Sampling)
```

```
Chain 2: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 2: Iteration: 500 / 500 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.717 seconds (Warm-up)
Chain 2:                0.66 seconds (Sampling)
Chain 2:                1.377 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000159 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.59 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: 0.731 seconds (Warm-up)
Chain 3:                0.527 seconds (Sampling)
Chain 3:                1.258 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000147 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.47 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: 0.75 seconds (Warm-up)
Chain 4:                0.623 seconds (Sampling)
Chain 4:                1.373 seconds (Total)
Chain 4:

Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#bulk-ess
```

The model's intercept (beta1) and the coefficients for gestational age (beta[2]) and prematurity (beta[3]) contribute to the prediction of log birth weight, with positive associations indicated by their mean estimates. In addition, the Rhat values are all close to or exactly 1, which typically suggests that the chains have converged.

The interaction term beta[4] has a mean estimate of 0.1969, which implies that the effect of gestational age on birth weight is different for preterm babies than for full-term babies. Specifically, the slope of the relationship is steeper for preterm births compared to full-term births, indicating that each additional week of gestation is associated with a greater increase in birth weight for preterm births than for full-term births. This result agrees with plot in Question 1.

```r
summary(mod2)$summary[c("beta[1]", "beta[2]", "beta[3]", "beta[4]", "sigma"),]
```

```
            mean      se_mean          sd       2.5%       25%       50%
beta[1] 1.1697984 6.988659e-05 0.002653556 1.16488936 1.1679383 1.1698020
beta[2] 0.1020085 1.031041e-04 0.003530353 0.09531607 0.0995568 0.1018874
beta[3] 0.5550383 3.332277e-03 0.066130187 0.43201346 0.5108913 0.5549890
beta[4] 0.1969413 6.963686e-04 0.013430650 0.17169772 0.1876603 0.1972734
sigma   0.1612443 8.144790e-05 0.001815965 0.15780069 0.1599945 0.1612466
             75%     97.5%     n_eff      Rhat
beta[1] 1.1717013 1.1749061 1441.6803 0.9968325
beta[2] 0.1044938 0.1088995 1172.4229 0.9983309
```

```
beta[3] 0.5977654 0.6834633  393.8377 1.0046655
beta[4] 0.2063210 0.2238342  371.9767 1.0058074
sigma   0.1625328 0.1647124  497.1130 1.0011154
```

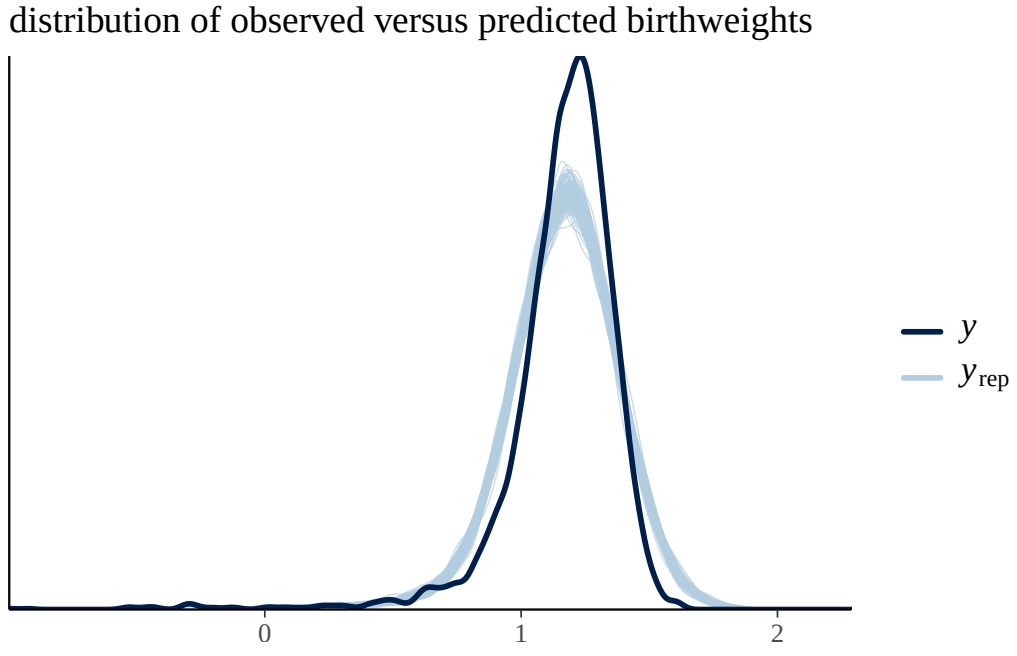
PPCs

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"]]
dim(yrep1)
```

```
[1] 1000 3842
```

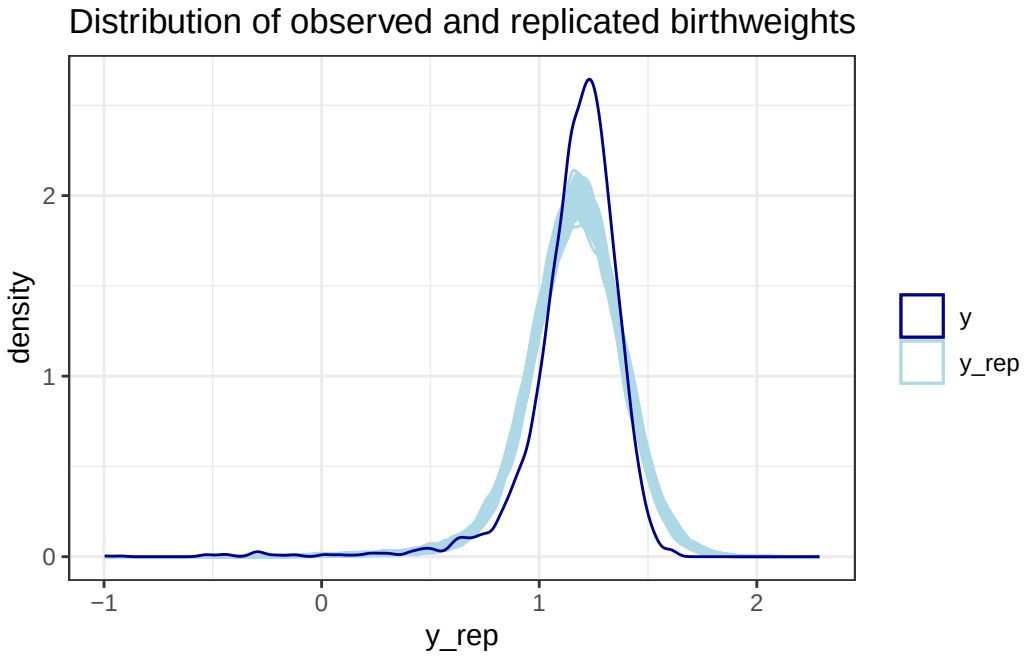
```
samp100 <- sample(nrow(yrep1), 100)
ppc_dens_overlay(y, yrep1[samp100, ])  + ggtitle("distribution of observed versus predicte
```



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))

# Turn into long format
drep <- drep |>
  pivot_longer(-(i:log_weight_obs), names_to = "sim", values_to ="y_rep")

# Filter to just include 100 draws and plot
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 and replicated birthweights") +
  theme_bw()
```



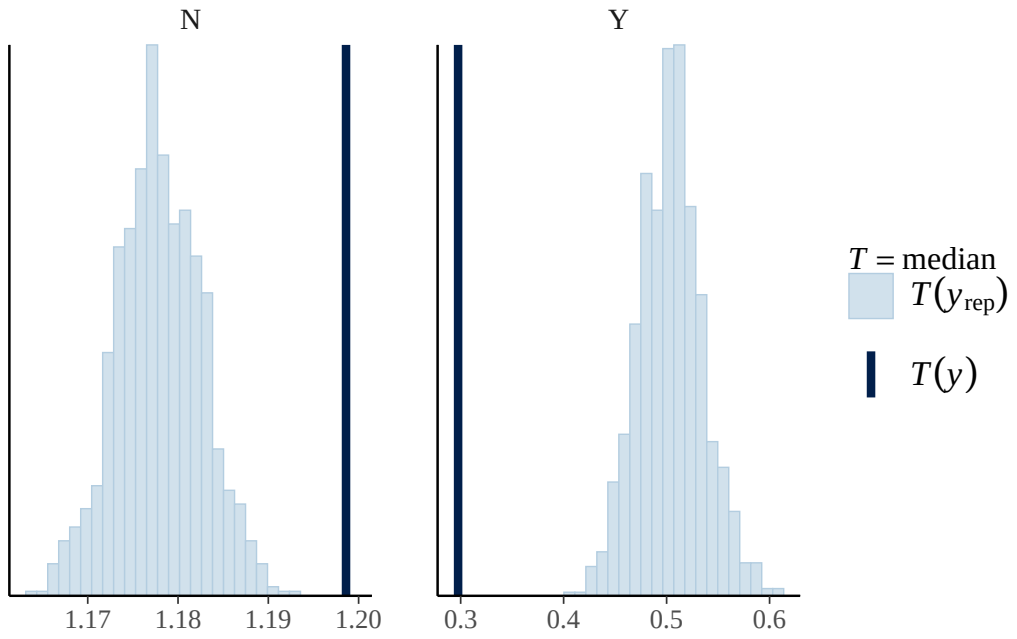
Test statistics

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')
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

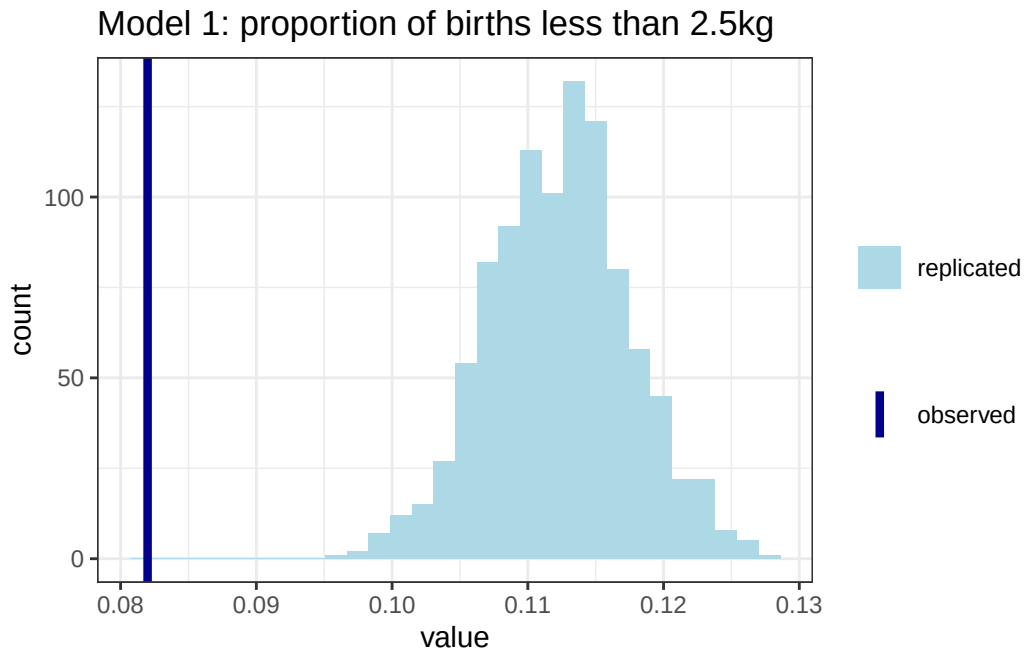


7.

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).

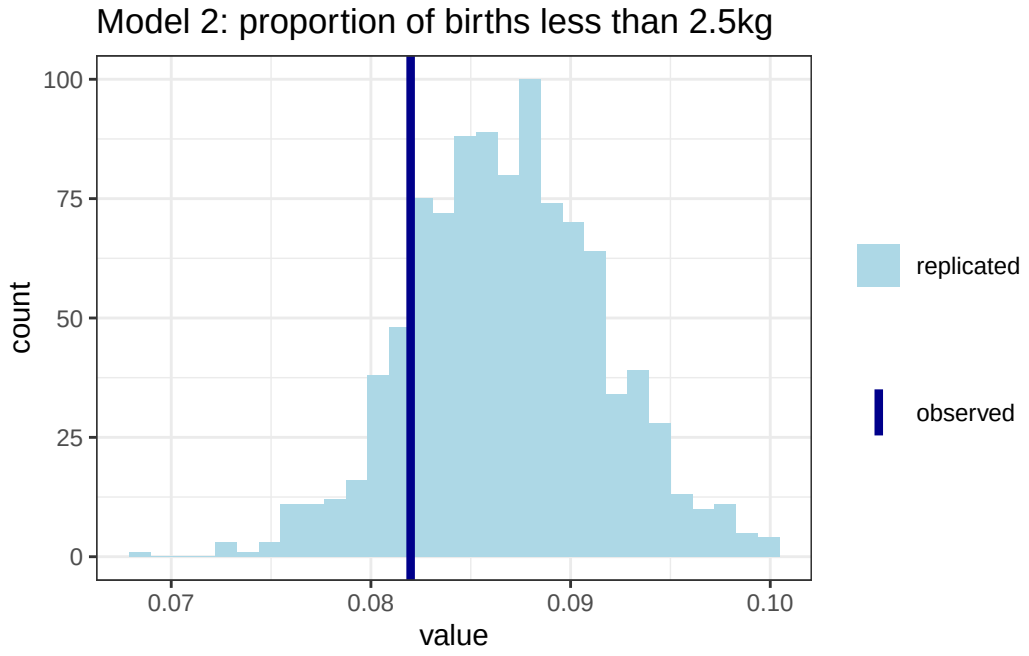
```
yrep2 <- extract(mod2)[["log_weight_rep"]]
test_y <- mean(y<=log(2.5))
test_y_rep <- sapply(1:nrow(yrep1), function(i) mean(yrep1[i,]<=log(2.5)))
test_y_rep_2 <- sapply(1:nrow(yrep2), function(i) mean(yrep2[i,]<=log(2.5)))

ggplot(data = as_tibble(test_y_rep), aes(value)) +
    geom_histogram(aes(fill = "replicated")) +
    geom_vline(aes(xintercept = test_y, color = "observed"), lwd = 1.5) +
  ggtitle("Model 1: proportion of births less than 2.5kg") +
  theme_bw() +
  scale_color_manual(name = "",
                     values = c("observed" = "darkblue"))+
  scale_fill_manual(name = "",
                     values = c("replicated" = "lightblue"))
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(data = as_tibble(test_y_rep_2), aes(value)) +
    geom_histogram(aes(fill = "replicated")) +
    geom_vline(aes(xintercept = test_y, color = "observed"), lwd = 1.5) +
  ggtitle("Model 2: proportion of births less than 2.5kg") +
  theme_bw() +
  scale_color_manual(name = "",
                     values = c("observed" = "darkblue"))+
  scale_fill_manual(name = "",
                     values = c("replicated" = "lightblue"))
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



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)
```

```
Warning: Relative effective sample sizes ('r_eff' argument) not specified.
For models fit with MCMC, the reported PSIS effective sample sizes and
MCSE estimates will be over-optimistic.
```

Look at the output:

```
loo1
```

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

         Estimate    SE
elpd_loo   1377.4  72.6
p_loo         9.3   1.4
looic     -2754.8 145.2
------
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.
```

8.

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

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

```
Warning: Relative effective sample sizes ('r_eff' argument) not specified.
For models fit with MCMC, the reported PSIS effective sample sizes and
MCSE estimates will be over-optimistic.

Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') f
```

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

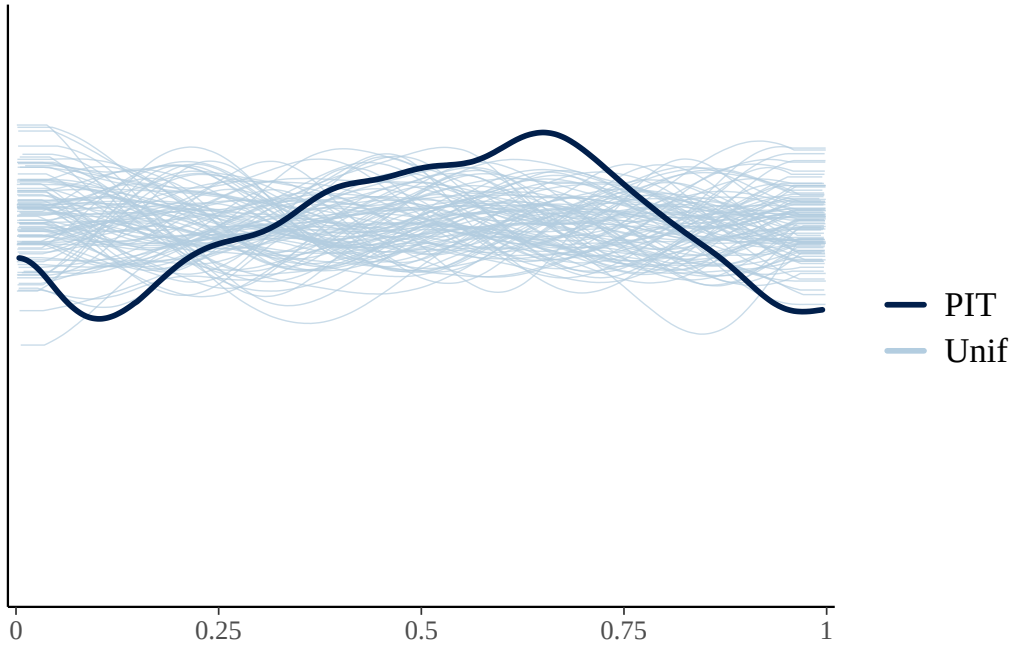
```
       elpd_diff se_diff
model2    0.0       0.0
model1 -175.1      36.4
```

When compared to Model 1, Model 2 has an elpd_diff of -175.1 with a standard error of the difference (se_diff) of 36.4. This indicates that Model 2 is expected to have a higher predictive performance by 175.1 log points on the Expected Log Predictive Density (ELPD) scale than Model 1, which is a substantial difference.

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

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

```
NOTE: The kernel density estimate assumes continuous observations and is not optimal for dis
```



Bonus question (not required)

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

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.

we add (centered and standardized) (log) bmi to model2.

```
ds$log_weight <- log(ds$birthweight)
ds$log_gest_c <- (log(ds$gest) - mean(log(ds$gest)))/sd(log(ds$gest))
ds$prematurity <- ifelse(ds$preterm=="Y", 1, 0)
ds$log_bmi_c <- (log(ds$bmi) - mean(log(ds$bmi)))/sd(log(ds$bmi))
# put into a list
```

```r
stan_data <- list(N = nrow(ds),
                  log_weight = ds$log_weight,
                  log_gest = ds$log_gest_c,
                  prematurity = ds$prematurity,
                  interac = ds$prematurity*ds$log_gest_c,
                  log_bmi = ds$log_bmi_c)
```

```r
mod3 <- stan(data = stan_data,
             file = "code/models/simple_weight_mod3.stan",
             iter = 500,
             seed = 243)
```

```
Warning in readLines(file, warn = TRUE): incomplete final line found on
'/Users/euijinbaek/STA2201/labs/code/models/simple_weight_mod3.stan'
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000333 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 3.33 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: 0.973 seconds (Warm-up)
Chain 1:                0.626 seconds (Sampling)
Chain 1:                1.599 seconds (Total)
Chain 1:
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000269 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.69 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: 0.796 seconds (Warm-up)
Chain 2:                0.682 seconds (Sampling)
Chain 2:                1.478 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000176 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.76 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: 0.846 seconds (Warm-up)
Chain 3:                0.794 seconds (Sampling)
Chain 3:                1.64 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000235 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.35 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: 0.918 seconds (Warm-up)
Chain 4:                0.71 seconds (Sampling)
Chain 4:                1.628 seconds (Total)
Chain 4:

Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#bulk-ess

Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tai
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#tail-ess
```

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

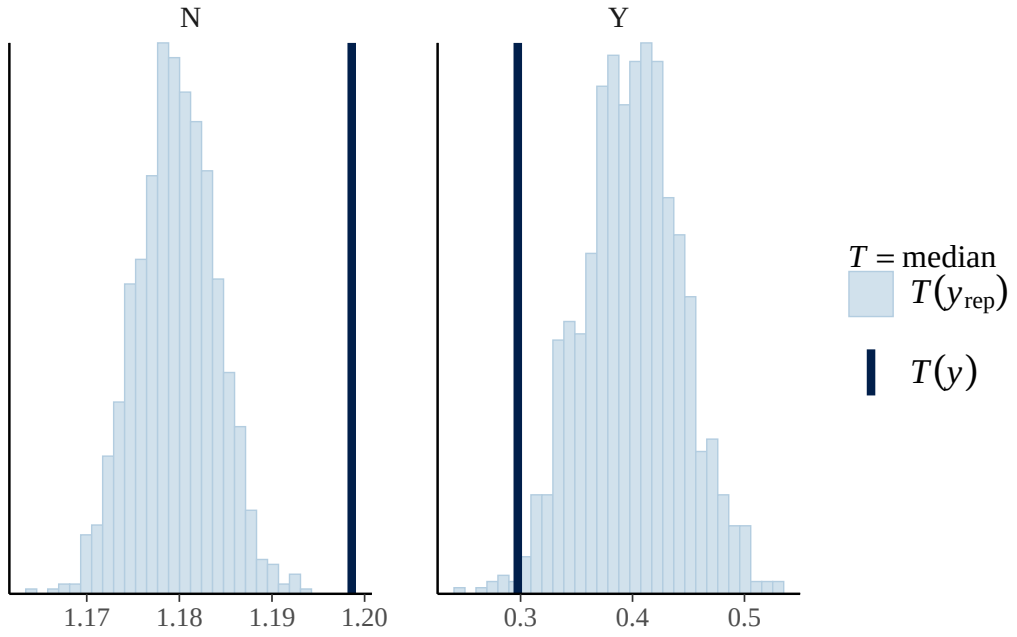
```
              mean      se_mean          sd        2.5%         25%         50%
beta[1] 1.16965871 6.883461e-05 0.002658302 1.164810586 1.167842930 1.16956710
beta[2] 0.10213438 1.243978e-04 0.003793301 0.095142208 0.099589242 0.10206743
beta[3] 0.57110212 4.083384e-03 0.064861306 0.440387626 0.526416370 0.57204832
beta[4] 0.20040149 8.115285e-04 0.013432858 0.173514003 0.191350230 0.20042302
beta[5] 0.01137366 6.803912e-05 0.002616623 0.006198954 0.009722757 0.01140201
sigma   0.16081858 7.997112e-05 0.001880679 0.157268733 0.159493499 0.16075843
              75%      97.5%     n_eff      Rhat
beta[1] 1.17143324 1.17476111 1491.4020 0.9970529
beta[2] 0.10474226 0.10948061  929.8425 1.0015079
beta[3] 0.61159306 0.70896015  252.3079 1.0249523
beta[4] 0.20925030 0.22778649  273.9866 1.0206622
beta[5] 0.01305821 0.01649379 1478.9886 1.0006557
sigma   0.16214344 0.16440446  553.0484 1.0062227
```

Then I checked the median by prematurity and proportion of births under 2.5kg.

```
set.seed(1856)
y <- ds$log_weight
yrep3 <- extract(mod3)[["log_weight_rep"]]
```

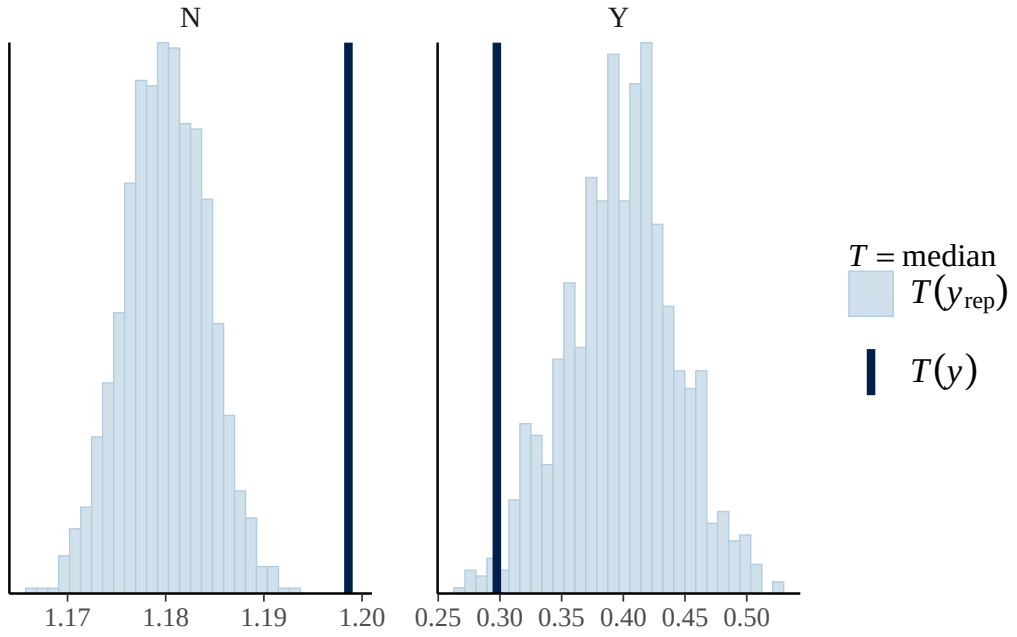
```
#model3
ppc_stat_grouped(ds$log_weight, yrep3, group = ds$preterm, stat = 'median')
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



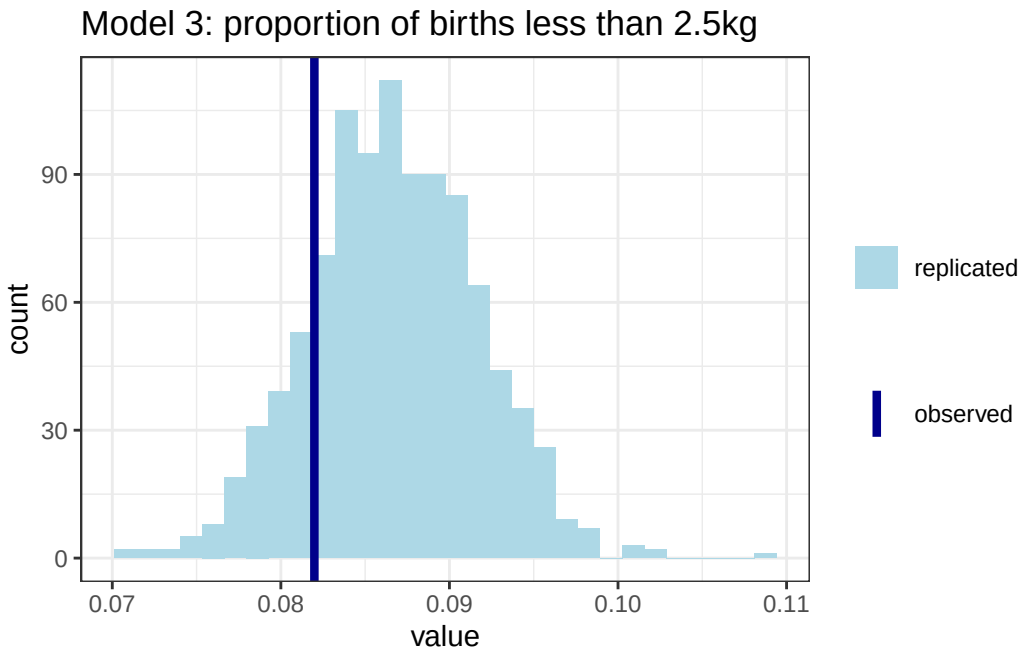

```
#model2
ppc_stat_grouped(ds$log_weight, yrep2, group = ds$preterm, stat = 'median')
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
test_y <- mean(y<=log(2.5))
test_y_rep_3 <- sapply(1:nrow(yrep3), function(i) mean(yrep3[i,]<=log(2.5)))
test_y_rep_2 <- sapply(1:nrow(yrep2), function(i) mean(yrep2[i,]<=log(2.5)))

ggplot(data = as_tibble(test_y_rep_3), aes(value)) +
    geom_histogram(aes(fill = "replicated")) +
    geom_vline(aes(xintercept = test_y, color = "observed"), lwd = 1.5) +
  ggtitle("Model 3: proportion of births less than 2.5kg") +
  theme_bw() +
  scale_color_manual(name = "",
                     values = c("observed" = "darkblue"))+
  scale_fill_manual(name = "",
                     values = c("replicated" = "lightblue"))
```

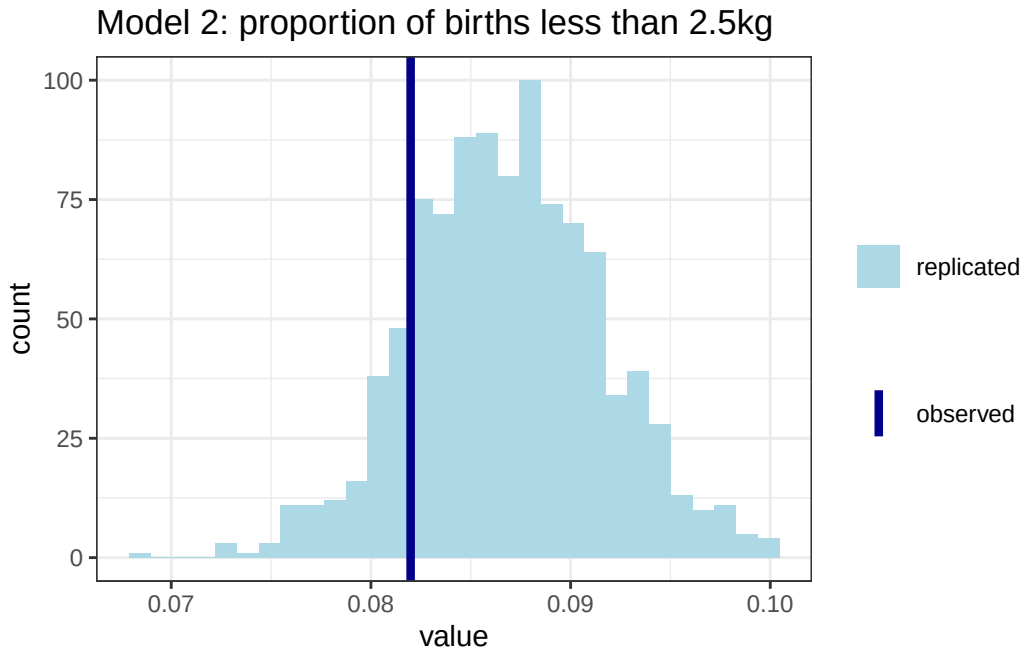
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(data = as_tibble(test_y_rep_2), aes(value)) +
    geom_histogram(aes(fill = "replicated")) +
    geom_vline(aes(xintercept = test_y, color = "observed"), lwd = 1.5) +
  ggtitle("Model 2: proportion of births less than 2.5kg") +
  theme_bw() +
```

```
    scale_color_manual(name = "",
                       values = c("observed" = "darkblue"))+
    scale_fill_manual(name = "",
                       values = c("replicated" = "lightblue"))
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



For both median by preterm and proportion of births under 2.5kg, the replicated values are both almost in the same location near observed value. Hence, there is no significant difference between model3 and model2.