Week 6: Visualizing the Bayesian Workflow

20/02/24

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)
library(skimr)
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.

```
skim(ds)
```

Table 1: Data summary

Name	ds
Number of rows	3842
Number of columns	9
_____	
Column type frequency:	
character	3
numeric	6
_____	
Group variables	None

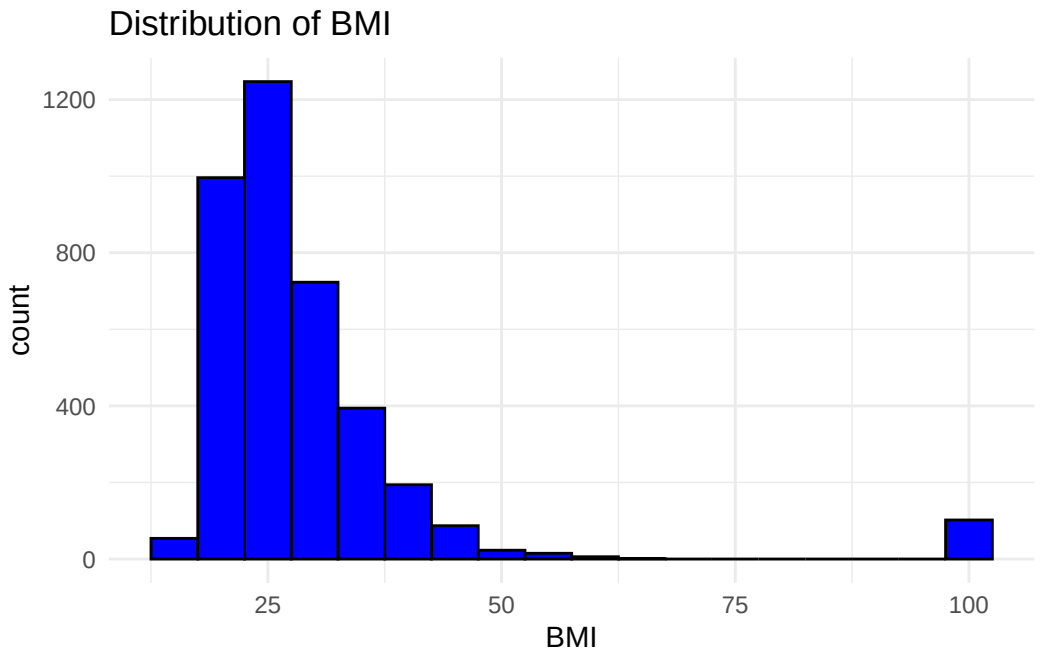
Variable type: character

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

Variable type: numeric

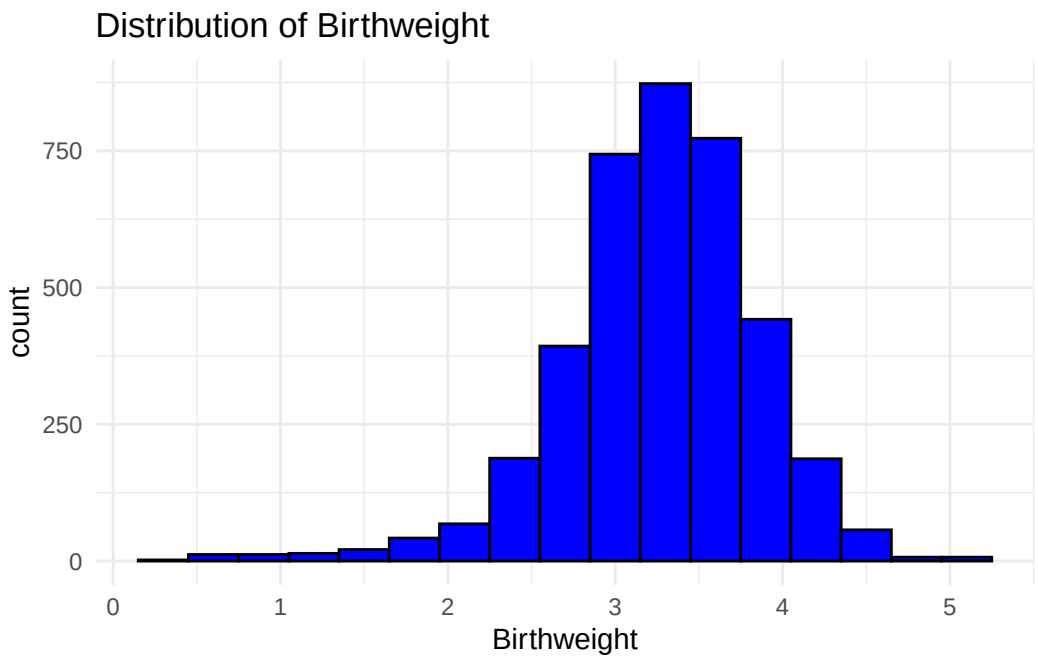
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
mager	0	1	28.93	5.84	14.00	25.00	29.0	33.00	50.00	
mracehisp	0	1	2.88	2.51	1.00	1.00	1.0	6.00	8.00	
meduc	0	1	4.41	1.81	1.00	3.00	4.0	6.00	9.00	
bmi	0	1	29.13	13.53	13.60	22.30	25.8	31.30	99.90	
gest	0	1	38.59	2.40	21.00	38.00	39.0	40.00	47.00	
birthweight	0	1	3.26	0.58	0.37	2.95	3.3	3.63	5.05	

```
ds|>
  ggplot(aes(bmi)) +
  geom_histogram(binwidth = 5, fill = "blue", color = "black") +
  labs(title = "Distribution of BMI", x = "BMI") +
  theme_minimal()
```



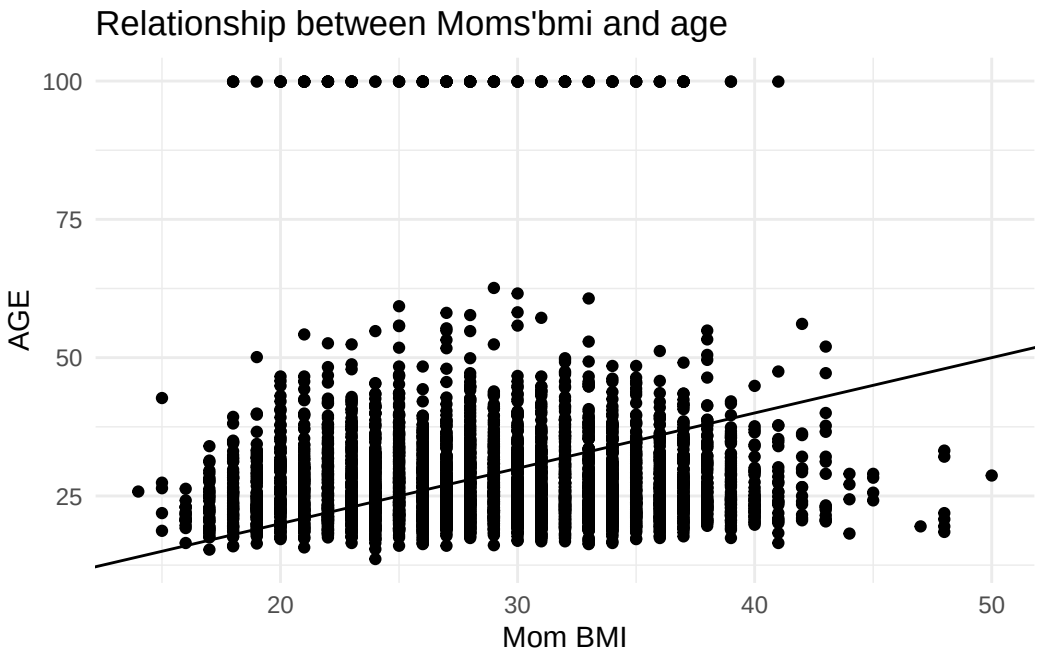
The histogram that represents the distribution of BMI of the moms shows a right-skewed distribution with the majority of BMI values concentrated between approximately 20 and 35. The highest frequency of BMI values, is in the range of approximately 22 to 30. The frequency of occurrences decreases sharply for BMI values above 30, and there are very few occurrences of BMI values above 50. There is also a small number of occurrences of extremely high BMI values, near 100, which appear to be outliers. The skewness of the distribution and highest frequency of BMI suggest that there is a higher concentration of mom with a BMI in the overweight range.

```
ds|>
  ggplot(aes(birthweight)) +
  geom_histogram(binwidth = 0.3, fill = "blue", color = "black") +
  labs(title = "Distribution of Birthweight", x = "Birthweight") +
  theme_minimal()
```



The plot is a histogram displaying the distribution of birthweight. The distribution appears to be roughly bell-shaped, centered around 3 to 4 kilograms, which is the most common range for birthweight, consistent to what we get in the table that mean is 3.264679 kg. The birthweight that has the highest frequency , is in the 3 to 3.5-kilogram range. The distribution has a slight right skew, since more outliers appear on right side (heavier birthweights) then on the left side (lighter birthweights).

```
ds|>
  ggplot(aes(mager,bmi)) +
  geom_point() +
  geom_abline()+
  labs(title = "Relationship between Moms'bmi and age", x = "Mom BMI", y = "AGE") +
  theme_minimal()
```



The scatter plot illustrates the relationship between the body mass index (BMI) of mothers and their age, data points spread widely, which suggests a considerable variability in age at any BMI levels.There is a line cross the scatter plot that appears to be sloping upwards, but not fit the dataset well, indicating that as the age of mothers increases, there is a general tendency for their bmi to increase as well. This could suggest a positive correlation between these two variables, meaning that higher BMI might be associated with older age in this dataset.

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).

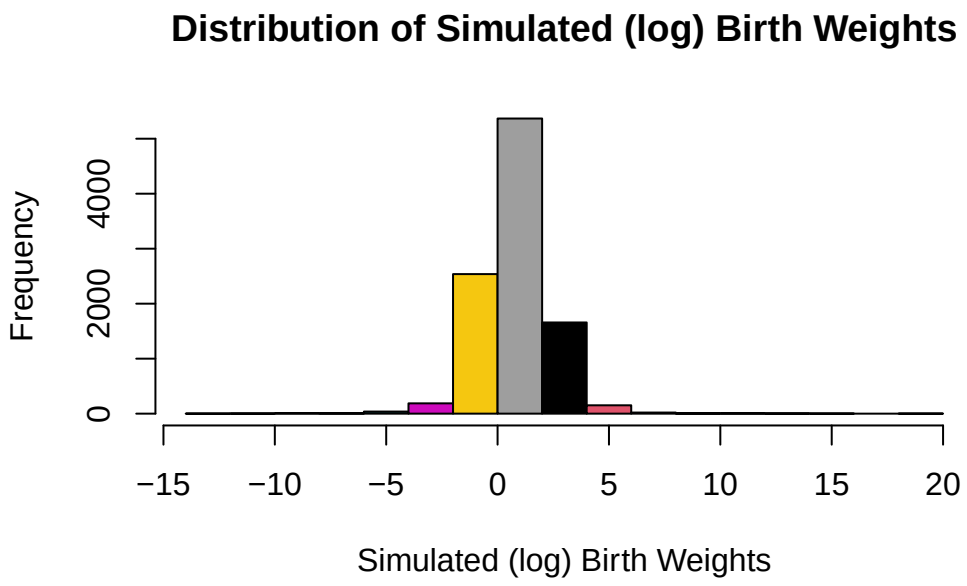
Question 2

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**.

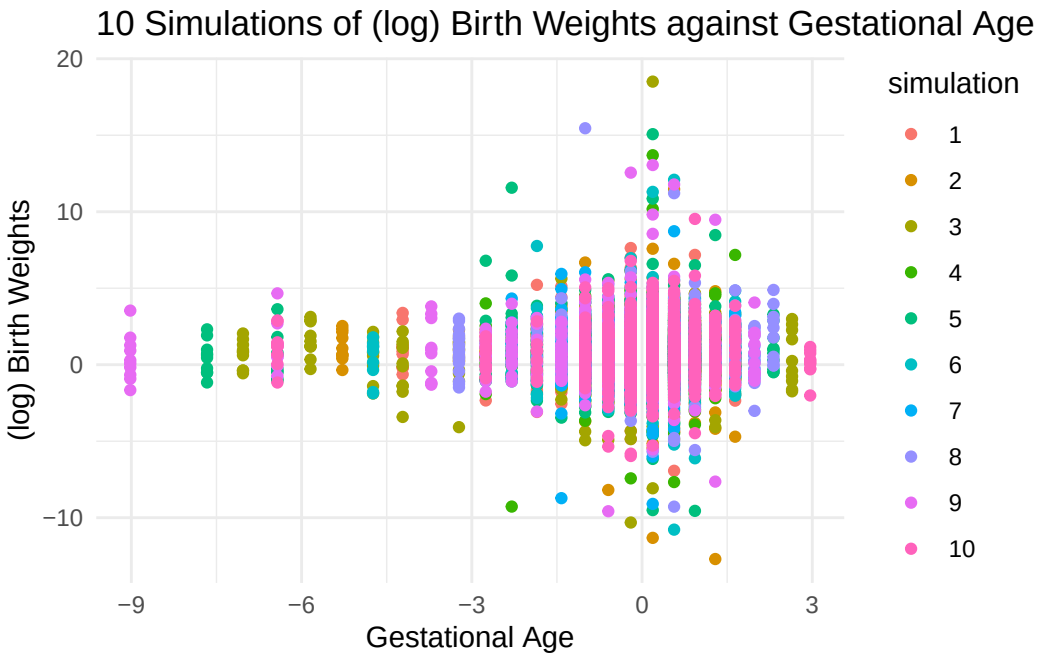
- Plot the resulting distribution of simulated (log) birth weights.
- Plot ten simulations of (log) birthweights against gestational age.

```
set.seed(100)
n<-1000
Y <- log(ds$gest)
Ysca <- scale(Y)
samp <- sample(nrow(Ysca),n)
beta1 <- matrix(nrow = n,ncol = 10)
beta2 <- matrix(nrow = n,ncol = 10)
sigma <- matrix(nrow = n,ncol = 10)
weight <- matrix(nrow = n,ncol = 10)


for (i in 1:10){
beta1[,i] <- rnorm(1000,mean = 0,sd=1)
```

```
beta2[,i] <- rnorm(1000,mean = 0,sd=1)
sigma[,i] <- abs(rnorm(1000,mean = 0,sd = 1))

weight[,i] <- beta1[,i] + beta2[,i]*Ysca[samp] + sigma[,i]
}


hist(weight, col = 1:10, main = "Distribution of Simulated (log) Birth Weights", xlab = "S
```




```
simulation <- data.frame(gestational_age = rep(Ysca[samp], each = 10),
                         weight = as.vector(weight),
                         sum = rep(1:10, each = 1000))


ggplot(simulation, aes(x = gestational_age, y = weight, color = factor(sum))) +
  geom_point() +
  labs(title = "10 Simulations of (log) Birth Weights against Gestational Age",
       x = "Gestational Age",
       y = "(log) Birth Weights",
       color = "simulation") +
  theme_minimal()
```



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)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: ‘Apple clang version 14.0.0 (clang-1400.0.29.202)’
using SDK: ‘MacOSX13.1.sdk’
clang -arch x86_64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/R
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/RcppEigen/include/Eigen
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/RcppEigen/include/Eigen
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/RcppEigen/include/Eigen
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000421 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 4.21 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: 0.476 seconds (Warm-up)
Chain 1:                0.377 seconds (Sampling)
Chain 1:                0.853 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000225 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.25 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: 0.456 seconds (Warm-up)
Chain 2:                0.38 seconds (Sampling)
Chain 2:                0.836 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000229 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.29 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: 0.464 seconds (Warm-up)
Chain 3:                0.392 seconds (Sampling)
Chain 3:                0.856 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000217 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.17 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: 0.454 seconds (Warm-up)
Chain 4:                0.396 seconds (Sampling)
Chain 4:                0.85 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 7.819072e-05 0.002741219 1.1569041 1.1608022 1.1626034
beta[2] 0.1437597 6.826879e-05 0.002727365 0.1381288 0.1418676 0.1437582
sigma   0.1689121 1.079225e-04 0.001918795 0.1651348 0.1677272 0.1689781
              75%     97.5%     n_eff      Rhat
beta[1] 1.1644381 1.1676158 1229.0716 0.9967375
beta[2] 0.1456099 0.1490155 1596.0337 0.9969379
sigma   0.1701247 0.1724931  316.1065 1.0038778
```

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.

```
fit1 <- extract(mod1)
beta_fit1 <- fit1$beta
new_gest <- log(abs(37-mean(ds$gest))/sd(ds$gest))
beta1_fit1 <- beta_fit1[,1]
beta2_fit1 <- beta_fit1[,2]
pred_weight <- median(beta1_fit1) + median(beta2_fit1)*new_gest
exp(pred_weight)
```

```
[1] 3.015177
```

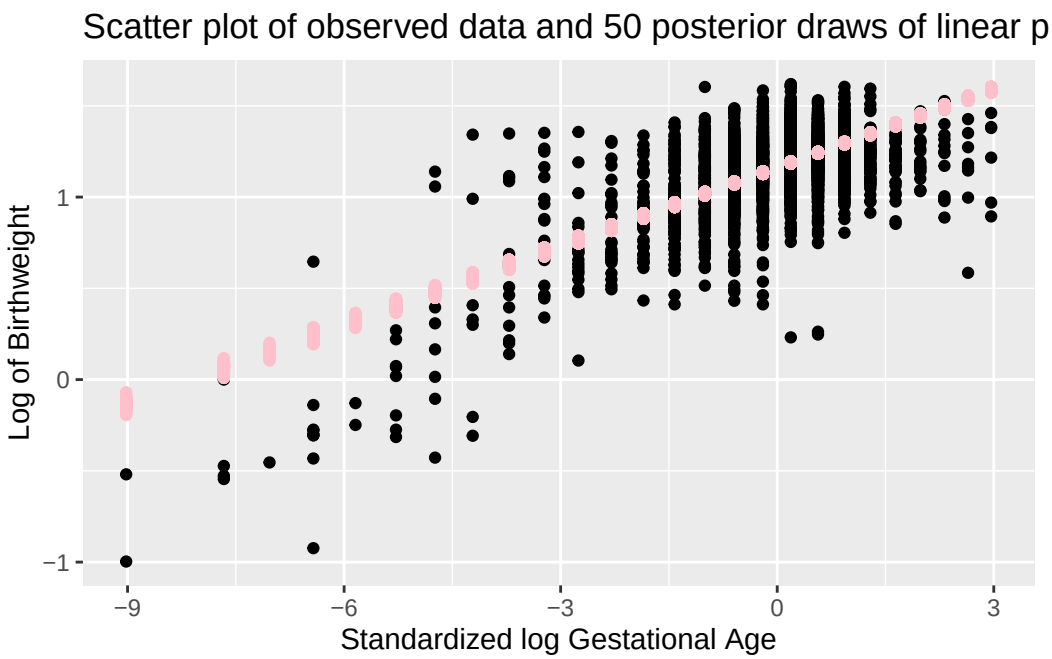
The estimated expected birthweight of a baby who was born at a gestational age of 37 weeks is 3.015177 kg, based on model 1.

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.

```
pred_matrix <- matrix(ncol = 3842,nrow=50)
for (i in 1:50){
  pred_matrix[i,] <- beta1_fit1[i] + beta2_fit1[i]*ds$log_gest_c
}
predictions <- data.frame(gestation = rep(ds$log_gest_c,each=50),log_weight_predicted = as
observations <- data.frame(gestation = ds$log_gest_c, log_weight_observed = ds$log_weight)

ggplot()+
  geom_point(data = observations,aes(x=gestation,y=log_weight_observed),col = 'black')+
```

```
geom_point(data = predictions, aes(x=gestation,y=log_weight_predicted),col = 'pink')+
labs(title = "Scatter plot of observed data and 50 posterior draws of linear predictor",
     x = "Standardized log Gestational Age",
     y = "Log of Birthweight")
```



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.

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


mod2 <- stan(data = stan_data,
             file = here("simple_weight_2.stan"),
             iter = 500,
```

```
            seed = 243)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: ‘Apple clang version 14.0.0 (clang-1400.0.29.202)’
using SDK: ‘MacOSX13.1.sdk’
clang -arch x86_64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/R
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/RcppEigen/include/Eigen
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/RcppEigen/include/Eigen
namespace Eigen {
              ^
              ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/RcppEigen/include/Eigen
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.001304 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 13.04 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: 0.951 seconds (Warm-up)
Chain 1:                0.832 seconds (Sampling)
Chain 1:                1.783 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000511 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 5.11 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: 1.128 seconds (Warm-up)
Chain 2:                0.99 seconds (Sampling)
Chain 2:                2.118 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000597 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 5.97 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: 1.041 seconds (Warm-up)
Chain 3:                0.865 seconds (Sampling)
Chain 3:                1.906 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000496 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 4.96 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: 1.041 seconds (Warm-up)
Chain 4:                0.885 seconds (Sampling)
Chain 4:                1.926 seconds (Total)
Chain 4:
```

```r
summary(mod2,pars=c("beta","sigma"))
```

```
$summary
             mean      se_mean          sd       2.5%       25%       50%
beta[1] 1.1702321 6.923871e-05 0.002579950 1.16538317 1.1684185 1.1701387
beta[2] 0.1032012 1.163022e-04 0.003595887 0.09599581 0.1007687 0.1032052
beta[3] 0.1097455 1.808195e-04 0.005048327 0.10059842 0.1061908 0.1095466
sigma   0.1622817 8.401150e-05 0.001759963 0.15861673 0.1611156 0.1623593
              75%     97.5%     n_eff      Rhat
beta[1] 1.1720559 1.1751970 1388.4323 0.9976248
beta[2] 0.1057029 0.1098795  955.9519 0.9991514
beta[3] 0.1134660 0.1191592  779.4791 0.9971717
sigma   0.1635318 0.1655079  438.8638 1.0024428

$c_summary
, , chains = chain:1

         stats
parameter      mean          sd       2.5%       25%       50%       75%
  beta[1] 1.1703592 0.002618671 1.16554914 1.1687733 1.1702370 1.1721670
  beta[2] 0.1033392 0.003519867 0.09661772 0.1006989 0.1034138 0.1058572
  beta[3] 0.1098605 0.005016212 0.10034662 0.1064255 0.1098467 0.1134067
  sigma   0.1624066 0.001774030 0.15916283 0.1611569 0.1624371 0.1636354
         stats
parameter     97.5%
  beta[1] 1.1754506
  beta[2] 0.1101387
  beta[3] 0.1197084
  sigma   0.1655644

, , chains = chain:2

         stats
parameter      mean          sd       2.5%       25%       50%       75%
  beta[1] 1.1701516 0.002443503 1.16543299 1.1682722 1.1700974 1.1719527
  beta[2] 0.1030311 0.003629490 0.09609565 0.1006864 0.1029622 0.1055175
  beta[3] 0.1098026 0.005158226 0.09888647 0.1064581 0.1097340 0.1133053
  sigma   0.1623411 0.001576135 0.15938720 0.1613677 0.1623389 0.1632460
         stats
parameter     97.5%
  beta[1] 1.1748909
  beta[2] 0.1095325
  beta[3] 0.1187928
  sigma   0.1655429
```

```
, , chains = chain:3

         stats
parameter      mean          sd      2.5%       25%       50%       75%
  beta[1] 1.1700945 0.002758143 1.1643263 1.1682834 1.1699427 1.1721205
  beta[2] 0.1033886 0.003589011 0.0959171 0.1008288 0.1034424 0.1057291
  beta[3] 0.1095124 0.005099645 0.1007348 0.1058179 0.1091010 0.1134668
  sigma   0.1621283 0.001975786 0.1581467 0.1609217 0.1621594 0.1634298
         stats
parameter     97.5%
  beta[1] 1.1749085
  beta[2] 0.1098771
  beta[3] 0.1190776
  sigma   0.1658467

, , chains = chain:4

         stats
parameter      mean          sd       2.5%       25%       50%       75%
  beta[1] 1.1703230 0.002493844 1.16591098 1.1685790 1.1702473 1.1718669
  beta[2] 0.1030459 0.003650496 0.09605465 0.1008762 0.1031220 0.1054614
  beta[3] 0.1098067 0.004939496 0.10085097 0.1061709 0.1093659 0.1137204
  sigma   0.1622510 0.001687693 0.15903105 0.1609125 0.1623701 0.1635669
         stats
parameter     97.5%
  beta[1] 1.1752186
  beta[2] 0.1094360
  beta[3] 0.1190288
  sigma   0.1651892
```

From the summary table, the coefficient of the interaction term is 0.11 approximately, which indicates that the impact of gestational age on birth weight is stronger for preterm than for non-preterm by 0.11. Thus the interpretation is that if baby is preterm birth, a one unit increase in log of gestational age in weeks, on average, associated with 0.11 units expected increase in log of birth weight.

PPCs

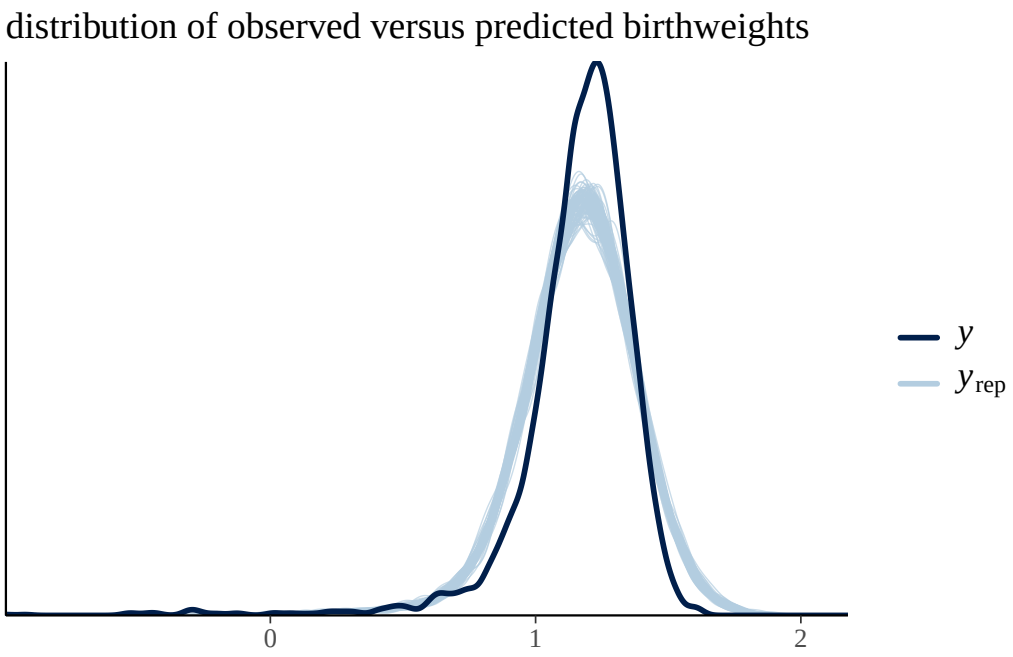
Now we've run two candidate models let's do some posterior predictive checks. The `bayesplot` package has a lot of inbuilt graphing functions to do this. For example, let's plot the distri-

bution 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
```

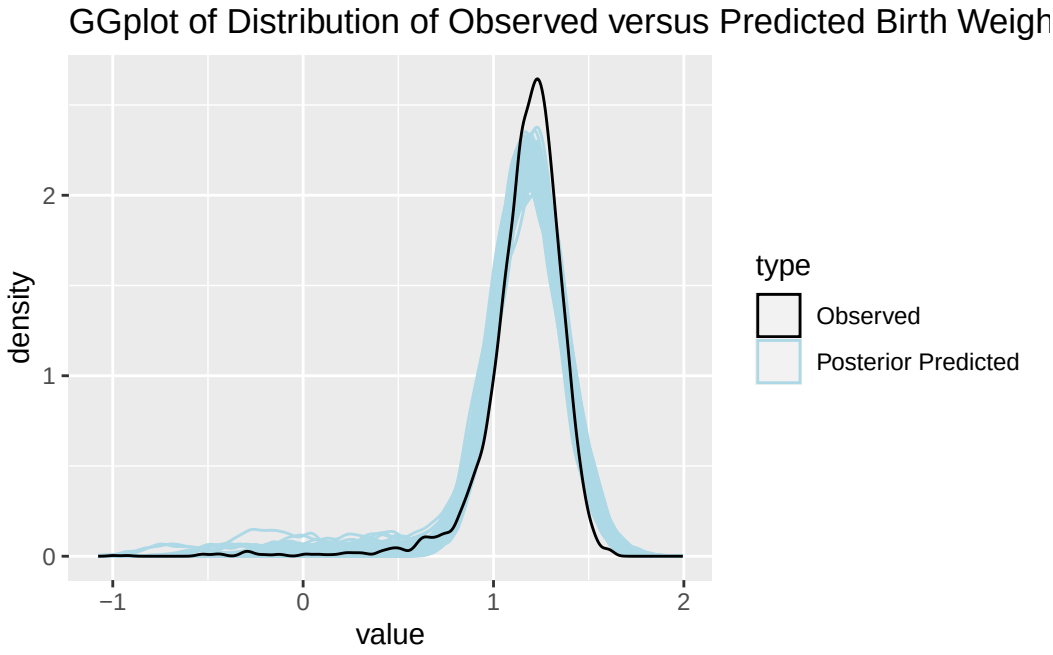


Question 6

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`)

```
fit2 <- extract(mod2)
yrep2 <- fit2$log_weight_rep
```

```r
weight_post <- yrep2[samp100,]

df_plot <- data.frame(
  value = c(as.vector(weight_post), y),
  type = rep(c(rep("Posterior Predicted", nrow(weight_post)), "Observed"), each = length(y
  draws = rep(0:nrow(weight_post), each = length(y))
)

ggplot(df_plot, aes(x = value, color = type, group = interaction(type, draws))) +
  geom_density() +
  scale_color_manual(values = c("black", "lightblue")) +
  ggtitle("GGplot of Distribution of Observed versus Predicted Birth Weights")
```

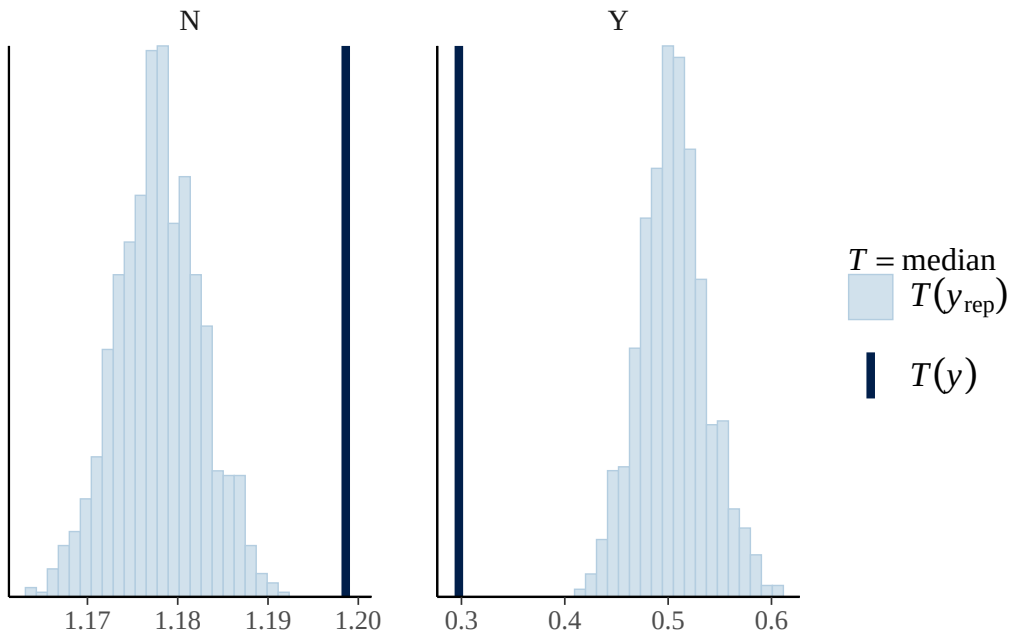


Test statistics

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

E.g. medians by prematurity for Model 1

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



Question 7

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

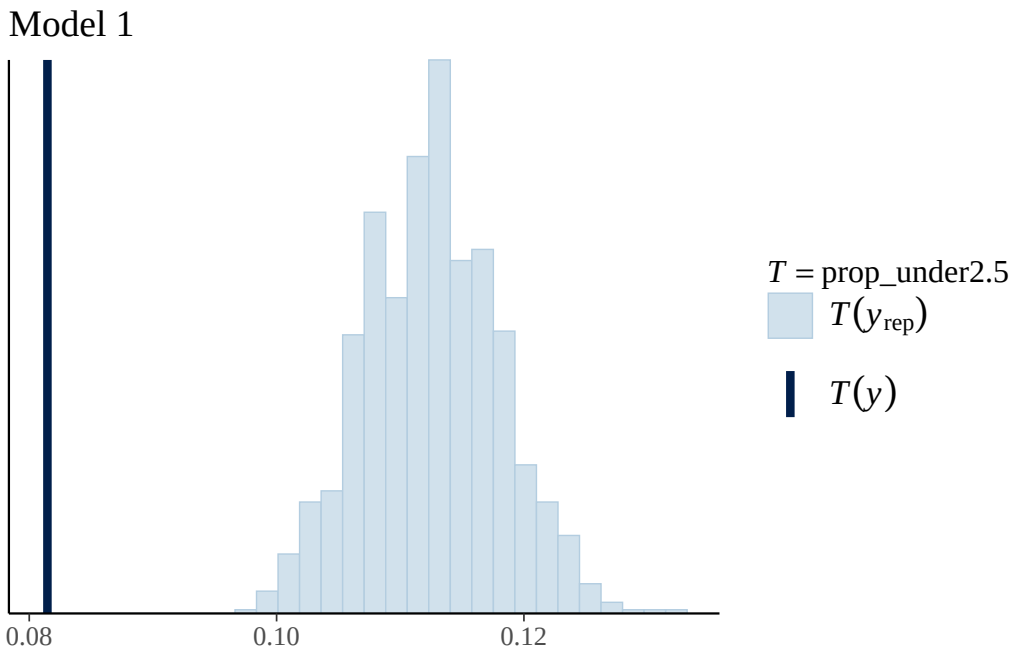
```
sum(ds$log_weight<log(2.5))/length(ds$log_weight)
```

```
[1] 0.08146799
```

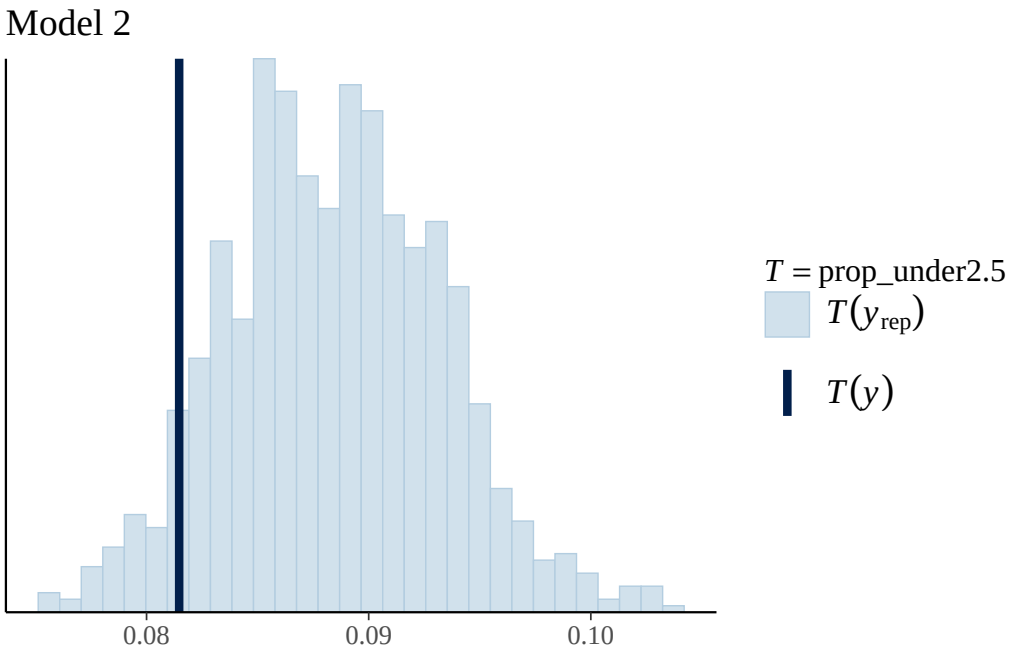
The test statistic of the proportion of births under 2.5kg is 0.08146799

```
count = 0
prop_under2.5 <- function(x){
for (i in 1:length(x))
  if(x[i]<log(2.5))
    count = count + 1
return(count/length(x))
```

```
}
post_fit1 <- fit1$log_weight_rep
post_fit2 <- fit2$log_weight_rep

ppc_stat(ds$log_weight,post_fit1,stat = 'prop_under2.5') + ggtitle("Model 1")
```




```
ppc_stat(ds$log_weight,post_fit2,stat = 'prop_under2.5') + ggtitle("Model 2")
```



From 2 plots, it is obvious that the model 2's proportion is closer to the the true proportion of births under 2.5kg compares to model 1, which suggests that the prediction performance of model 2 on this dataset is significantly better than that of model 1.

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
loglik2 <- fit2$log_lik
loo2 <- loo(loglik2, save_psis = TRUE)
loo_compare(loo1,loo2)
```

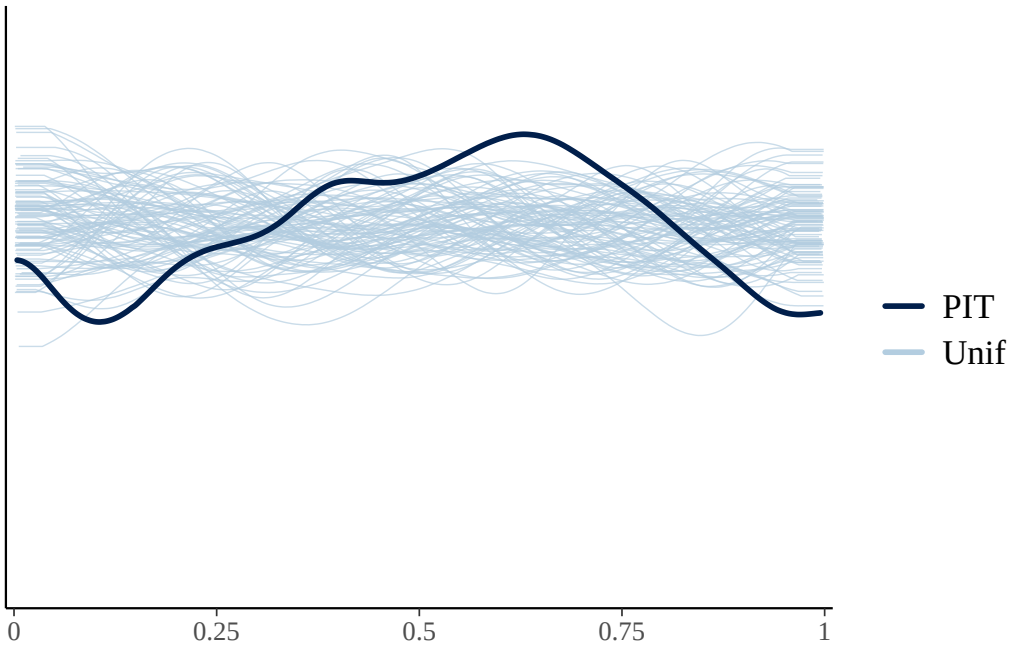
```
       elpd_diff se_diff
model2    0.0       0.0
model1 -152.8      35.3
```

From results of `loo_compare` function, suggesting that model 1 has a lower expected log predictive density compared to model 2 by 152.8 on the log scale and the standard error of model2 is 35.3 higher than model 1. Thus, model 2 appears to be a better fit to this dataset than model 1 since the elpd of model 2 is higher

LOO-PIT

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

```r
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.

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


mod3 <- stan(data = stan_data,
             file = here("simple_weight_3.stan"),
```

```
              iter = 500,
              seed = 243)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: 'Apple clang version 14.0.0 (clang-1400.0.29.202)'
using SDK: 'MacOSX13.1.sdk'
clang -arch x86_64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/R
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/RcppEigen/include/Eigen
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/RcppEigen/include/Eigen
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/RcppEigen/include/Eigen
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.005882 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 58.82 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: 5.38 seconds (Warm-up)
Chain 1:                7.155 seconds (Sampling)
Chain 1:                12.535 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000679 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 6.79 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: 4.752 seconds (Warm-up)
Chain 2:                5.421 seconds (Sampling)
Chain 2:                10.173 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000695 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 6.95 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:   1 / 500 [  0%]  (Warmup)
```

```
Chain 3: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 3: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 3: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 3: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 3: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 3: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 3: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 3: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 3: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 3: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 3: Iteration: 500 / 500 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 4.788 seconds (Warm-up)
Chain 3:                4.892 seconds (Sampling)
Chain 3:                9.68 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000731 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 7.31 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: 5.682 seconds (Warm-up)
Chain 4:                5.229 seconds (Sampling)
Chain 4:                10.911 seconds (Total)
Chain 4:
```

We first compare model 2 and 3 on LOO

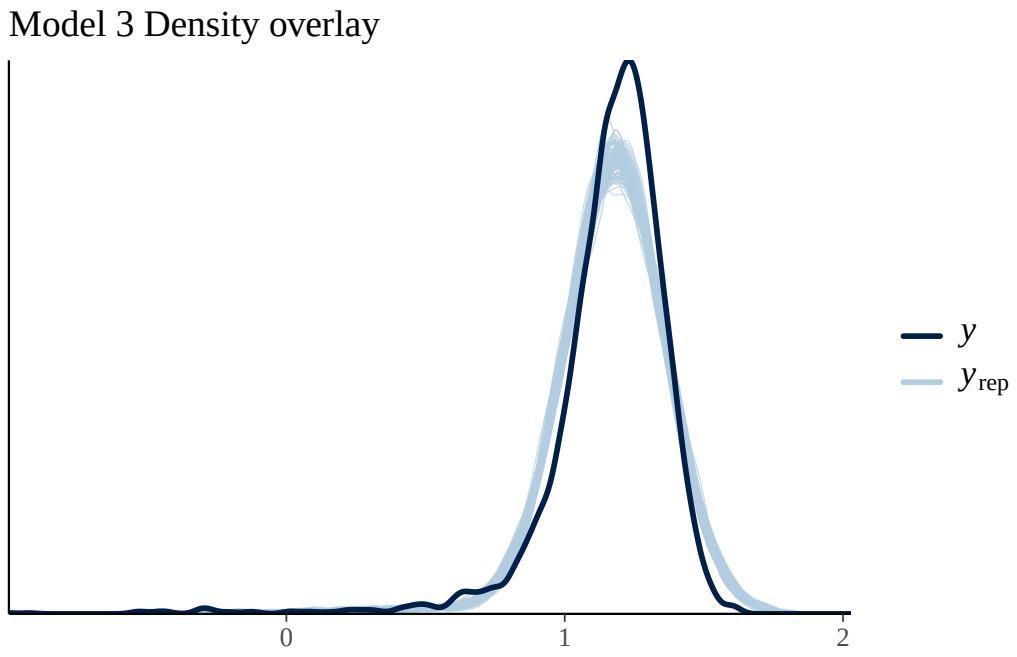
```
fit3 <- extract(mod3)
```

```
loglik3 <- fit3$log_lik
loo3 <- loo(loglik3,save_psis=TRUE)
loo_compare(loo2,loo3)
```

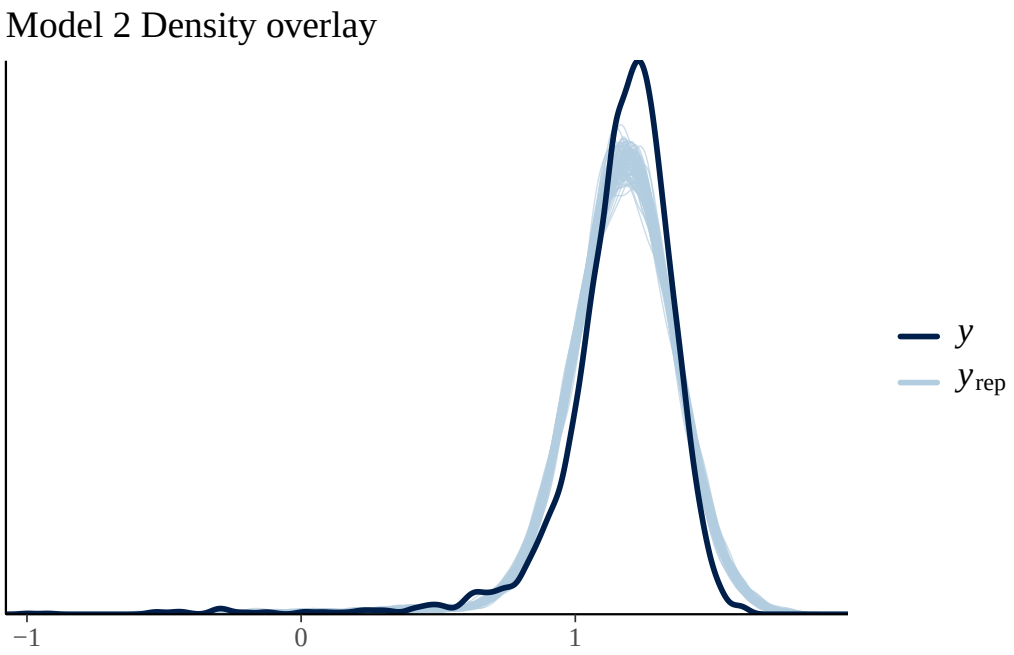
```
       elpd_diff se_diff
model2  0.0       0.0
model1 -7.1       4.3
```

then move to compare the density overlays of both models

```
yrep3 <- fit3$log_weight_rep
ppc_dens_overlay(y,yrep3[samp100,]) +ggtitle("Model 3 Density overlay")
```




```
ppc_dens_overlay(y,yrep2[samp100,]) +ggtitle("Model 2 Density overlay")
```



I add the new variable on the model which is the log of bmi. From density overlays of both models, we can find they are very similar. By loo_compare(), we observe that model 3 has higer elpd compares to model 2, thus the model 3 has better predictive performance compares with model2.