# Week 6 Lab Submission

Yeonjoon Choi

2023-02-21

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)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.4.0     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.2

## Warning: package 'ggplot2' was built under R version 4.2.2

## Warning: package 'tibble' was built under R version 4.2.2

## Warning: package 'dplyr' was built under R version 4.2.2

## Warning: package 'forcats' was built under R version 4.2.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(here)
```

```
## Warning: package 'here' was built under R version 4.2.2

## here() starts at C:/Users/choip/OneDrive/Documents/GitHub/STA2201H_Yeonjoon_Choi
```

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

```
## Loading required package: StanHeaders
```

```
##
## rstan version 2.26.13 (Stan version 2.26.1)
```

```
## 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)
```

```
## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
```

```
##
## Attaching package: 'rstan'
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```r
library(bayesplot)
```

```
## Warning: package 'bayesplot' was built under R version 4.2.2
```

```
## This is bayesplot version 1.10.0
```

```
## - 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
```

```r
library(loo)
```

```
## Warning: package 'loo' was built under R version 4.2.2
```

```
## This is loo version 2.5.1
```

```
## - 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 'cores' ar
```

```
## - Windows 10 users: loo may be very slow if 'mc.cores' is set in your .Rprofile file (see https://gi

##
## Attaching package: 'loo'

## The following object is masked from 'package:rstan':
##
##     loo
```

```
library(tidybayes)
require(curl)
```

```
## Loading required package: curl

## Using libcurl 7.64.1 with Schannel

##
## Attaching package: 'curl'

## The following object is masked from 'package:readr':
##
##     parse_date
```

```
gurl = "https://raw.githubusercontent.com/MJAlexander/applied-stats-2023/main/data/births_2017_sample.R
ds = readRDS(url(gurl, method="libcurl"))
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

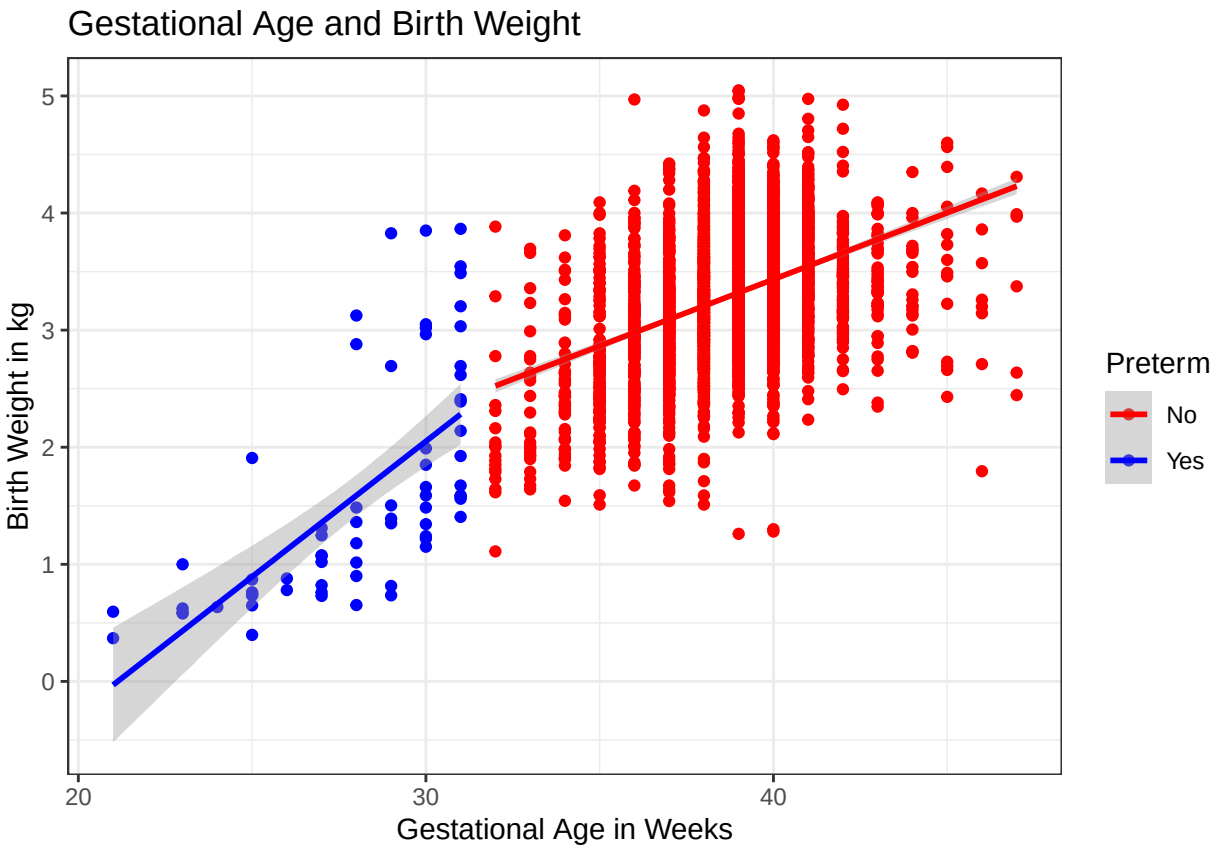
Use plots or tables to show three interesting observations about the data. Remember:

- 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(data = ds, aes(y = birthweight, x = gest, color = preterm))+
  geom_point()+
  geom_smooth(method = "lm")+
  xlab("Gestational Age in Weeks")+
  ylab("Birth Weight in kg")+
  labs(title = "Gestational Age and Birth Weight")+
  scale_color_manual(name="Preterm",
                     labels=c("No","Yes"),
                     values=c("red","blue"))+
  theme_bw()
```

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




From the plot, we see clear increase in birth weight as there is increase in gestational age. We also suspect that there is difference in effect of gestational age on birth weight, depending on whether or not the birth was preterm. The plot suggests that there is a possible interaction effect by the preterm variable.

We also note that there are less data points for preterm babies.

From the CDC's webpage, BMI of over 30 is defined as obesity: https://www.cdc.gov/obesity/basics/adult-defining.html

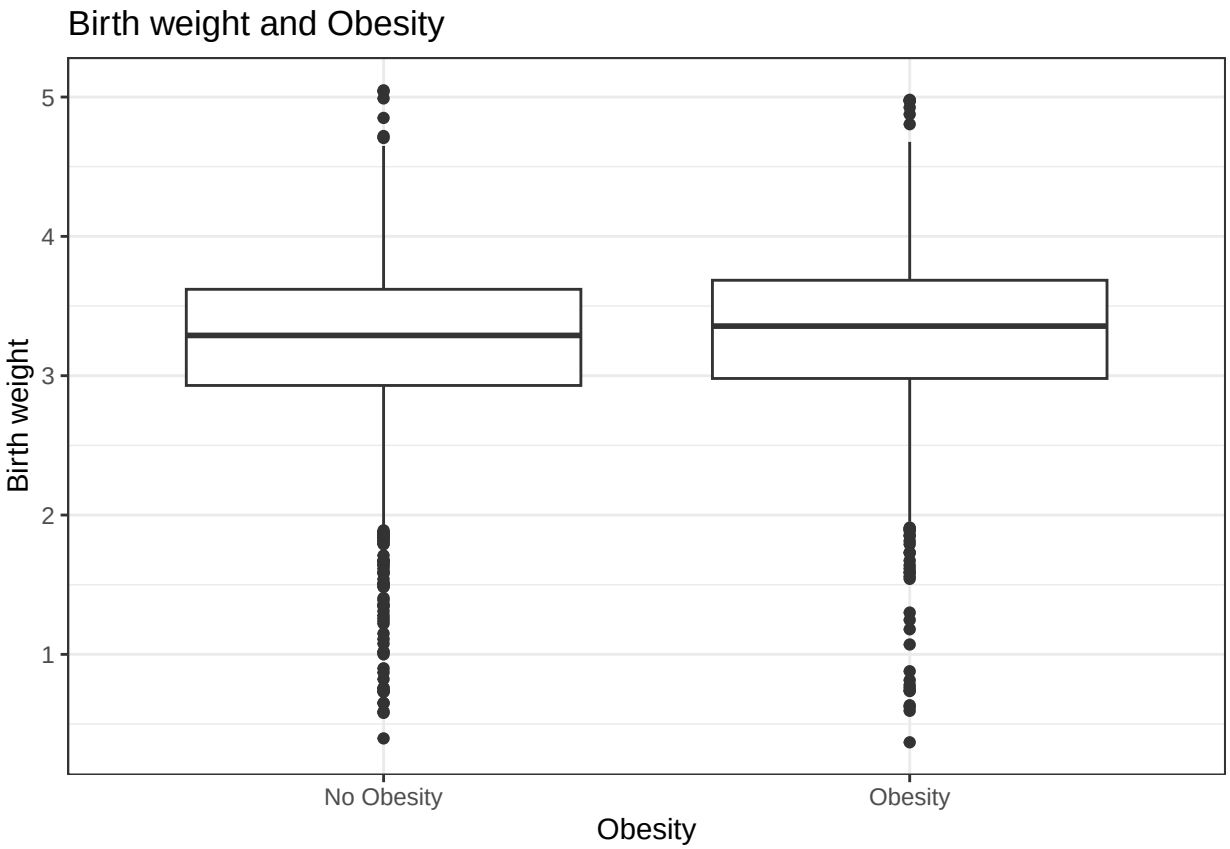
We add a categorical variable for obesity, with the variable being Y if the mother's BMI was over 30, and N otherwise.

```
ds = ds|>
  mutate(Obesity=ifelse(bmi<30, "N", "Y"))
```

There are 1182 observations with Obesity equal to Y, and 2660 with Obesity equal to N.

We plot a boxplot for baby's birth weight, divded into obesity group and non-obesity group.

```
ggplot(ds, aes(x=Obesity, y=birthweight)) +
  geom_boxplot()+
  xlab("Obesity")+
  ylab("Birth weight")+
  labs(title = "Birth weight and Obesity")+
  scale_x_discrete(labels = c("No Obesity", "Obesity"))+
  theme_bw()
```




In the boxplot, there is a small difference in the mean of obesity group and non-obesity group. We perform a t-test to see if the difference in mean is significant.

We also perform Mann–Whitney U test, which is a non-parametric test where the null hypothesis is that two groups come from the same distribution, as there are cases where t-test may fail due to the underlying distribution of the data.

```
require(pander)
```

```
## Loading required package: pander
```

```
pander(t.test(birthweight~Obesity, data = ds, conf.int = TRUE))
```

Table 1: Welch Two Sample t-test: `birthweight` by `Obesity` (continued below)

| Test statistic | df | P value | Alternative hypothesis |
|---|---|---|---|
| -3.068 | 2105 | 0.002185 * * | two.sided |

| mean in group N | mean in group Y |
|---|---|
| 3.245 | 3.309 |

```
pander(wilcox.test(birthweight~Obesity, data = ds, conf.int=TRUE))
```

Table 3: Wilcoxon rank sum test with continuity correction: `birthweight` by `Obesity` (continued below)

| Test statistic | P value | Alternative hypothesis |
|---|---|---|
| 1459226 | 0.0003767 * * * | two.sided |

| difference in location |
|---|
| -0.06602 |

So we do see that the difference in mean is significant, in both t-test and Mann–Whitney U test.

There has been studies showing some relations between low birth weight risk and race. In particular, these studies claimed that African American women were more likley to have babies with low birth weight.

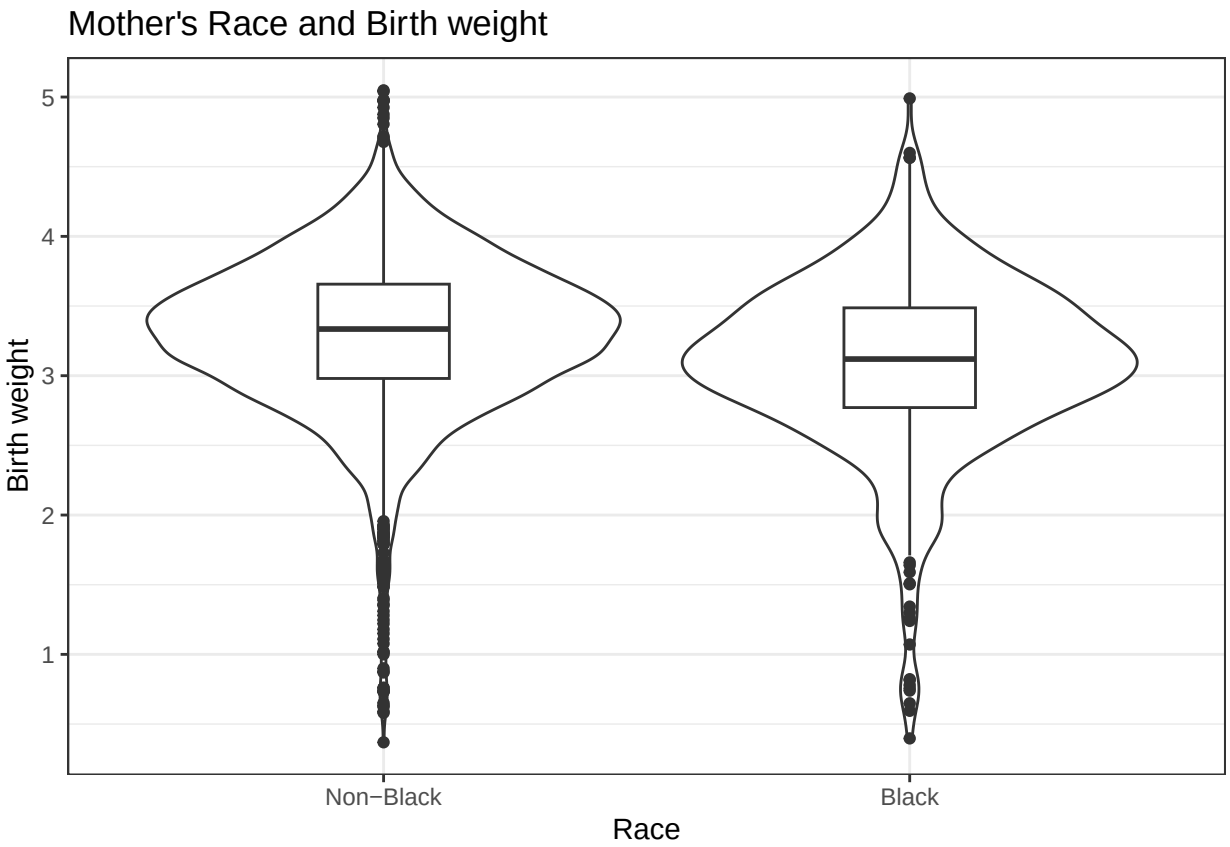
-https://mhnpjournal.biomedcentral.com/articles/10.1186/s40748-018-0084-2

-https://www.nejm.org/doi/full/10.1056/nejm199710233371706

(Note that first study used data from 2005-2014, and the second study used data from 1980-1995, while we are using data from 2017. We are not claiming definitive link between race and low birth weight, but we believe it is still worth investigating.)

We plot violin plot for the birth weight, for black mothers and non-black mothers. There were 574 black mothers. Violin plot shows the probability density of the data, fitted using kernel density estimators. Inside the violin plot, we plot a boxplot to show the mean and the interquartile range.

```
ds = ds|>
  mutate(Black = ifelse(mracehisp ==2 , "Y", "N") )


ggplot(ds, aes(x=Black, y=birthweight)) +
  geom_violin()+
  geom_boxplot(width=0.25)+
  xlab("Race")+
  ylab("Birth weight")+
  labs(title = "Mother's Race and Birth weight")+
  scale_x_discrete(labels = c("Non-Black", "Black"))+
  theme_bw()
```




From the plot, we notice that the mean birth weight is higher for non-black group, with the density peaking at a higher birth weight compared to the black group. We also notice that the tail end for the black group for low birth weight is thicker compared to the non-black group. However, this may be due to lower sample size in the black group.

We can perform t-test and Mann–Whitney U test like before.

```
pander(t.test(birthweight~Black, data = ds, conf.int = TRUE))
```

Table 5: Welch Two Sample t-test: `birthweight` by `Black` (continued below)

| Test statistic | df | P value | Alternative hypothesis |
|---|---|---|---|
| 7.64 | 740.4 | 6.736e-14 * * * | two.sided |

| mean in group N | mean in group Y |
|---|---|
| 3.297 | 3.079 |

```
pander(wilcox.test(birthweight~Black, data = ds, conf.int=TRUE))
```

Table 7: Wilcoxon rank sum test with continuity correction: `birthweight` by `Black` (continued below)

| Test statistic | P value | Alternative hypothesis |
|---|---|---|
| 1138887 | 2.411e-16 * * * | two.sided |

| difference in location |
|---|
| 0.203 |

Both test agrees that the difference is significant.

From WHO, low birth weight is defined as weight at birth less than 2.5 kg: https://www.who.int/data/nutrition/nlis/info/low-birth-weight

We add a column for low birth weight, with the variable being Y if the baby weighed less than 2.5kg, and N otherwise.

```
ds = ds|>
  mutate(low.birth = ifelse(birthweight<2.5, "Y", "N") )
```

There are 313 cases of low birth weight.

We want to see if Black Americans are more at risk for low birth weight.

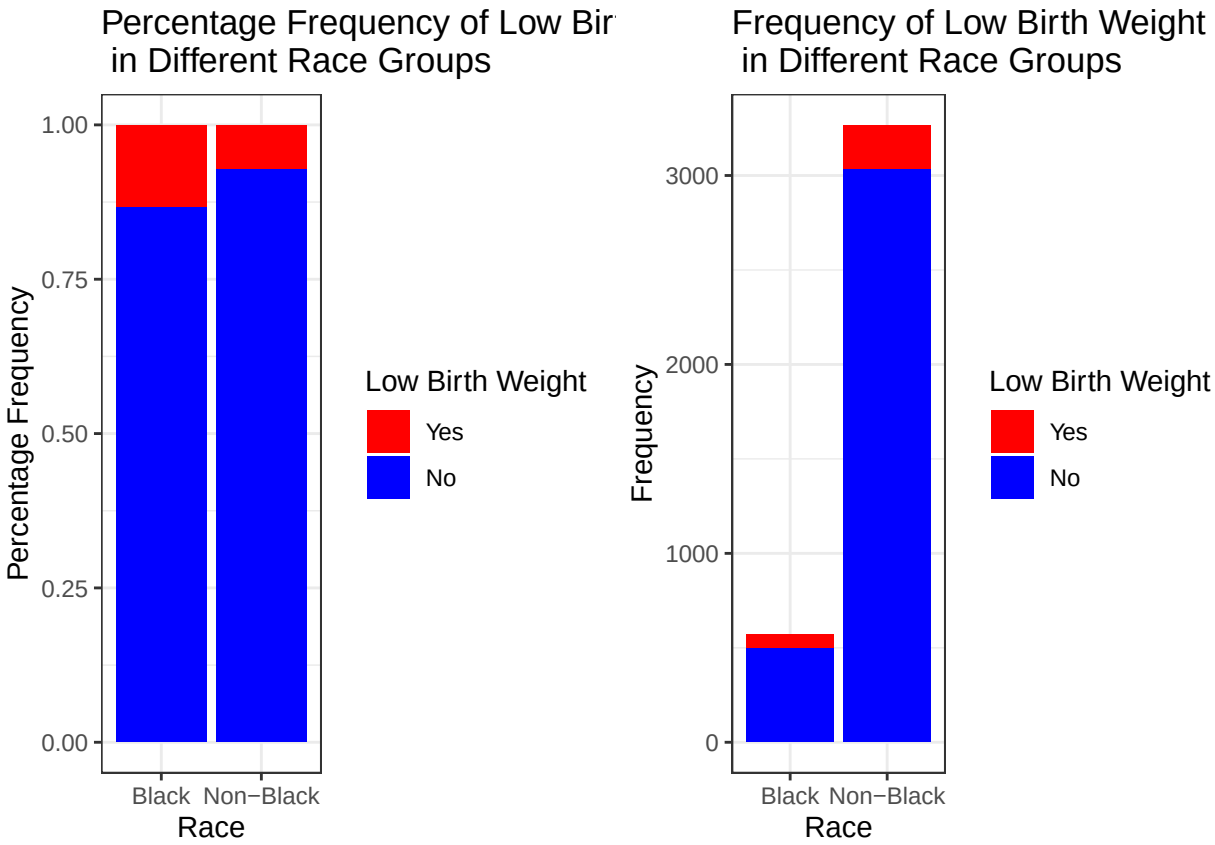
```
temp = table(ifelse(ds$Black == "Y", "Black", "Non-Black"),
             ifelse(ds$low.birth == "Y", "Low Birth Weight", "No Low Birth Weight"))

temp_2 = as.data.frame.table(temp)
p1 = ggplot(temp_2, aes(fill=Var2, y=Freq, x=Var1)) +
    geom_col(position = "fill")+
     scale_fill_manual(name="Low Birth Weight",
                       labels=c("Yes","No"),
                       values=c("red","blue"))+
    xlab("Race")+
    ylab("Percentage Frequency")+
    labs(title = "Percentage Frequency of Low Birth Weight \n in Different Race Groups")+
    theme_bw()

p2 = ggplot(temp_2, aes(fill=Var2, y=Freq, x=Var1)) +
    geom_bar(stat = "identity")+
   scale_fill_manual(name="Low Birth Weight",
                       labels=c("Yes","No"),
                       values=c("red","blue"))+
    xlab("Race")+
    ylab("Frequency")+
    labs(title = "Frequency of Low Birth Weight\n in Different Race Groups")+
    theme_bw()

require(gridExtra)
```

```
## Loading required package: gridExtra
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
grid.arrange(p1,p2, ncol = 2)
```




We see that in percentage, low birth weight happened more frequently for Black Americans.

We print out the two-way table for race and low birth weight.

```
temp = table(ifelse(ds$Black == "Y", "Black", "Non-Black"),
             ifelse(ds$low.birth == "Y", "Low Birth Weight", "No Low Birth Weight"))


require(knitr)
```

```
## Loading required package: knitr
```

```
kable(temp)
```

| | Low Birth Weight | No Low Birth Weight |
|---|---|---|
| Black | 77 | 497 |
| Non-Black | 236 | 3032 |

We test if the proportion of low birth weight among the two race group are the same.

```
pander(prop.test(x = c(77, 236), n = c(574, 3268), correct = FALSE))
```

Table 10: 2-sample test for equality of proportions without continuity correction: `c(77, 236) out of c(574, 3268)`

| Test statistic | df | P value | Alternative hypothesis | prop 1 | prop 2 |
|---|---|---|---|---|---|
| 25.02 | 1 | 5.66e-07 * * * | two.sided | 0.1341 | 0.07222 |

From the test result, we see that the proportion of low birth weight in Black Americans is significantly higher compared to other race group.

From this, we see that race may play a role in decrease in birth weight. In particular, it seems that Black Americans are more likely to have low birth weight compared to other races.

## 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 $\beta$s

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

and for $\sigma$

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

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).

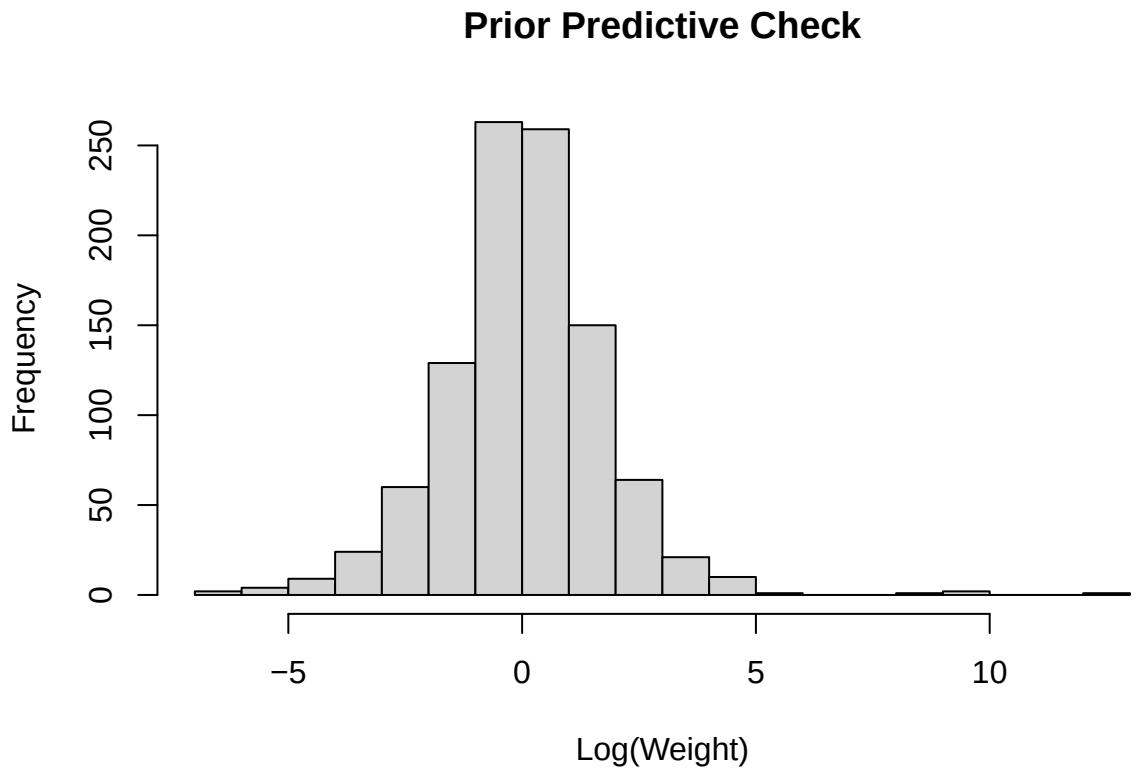
## Question 2

For Model 1, simulate values of $\beta$s and $\sigma$ 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.

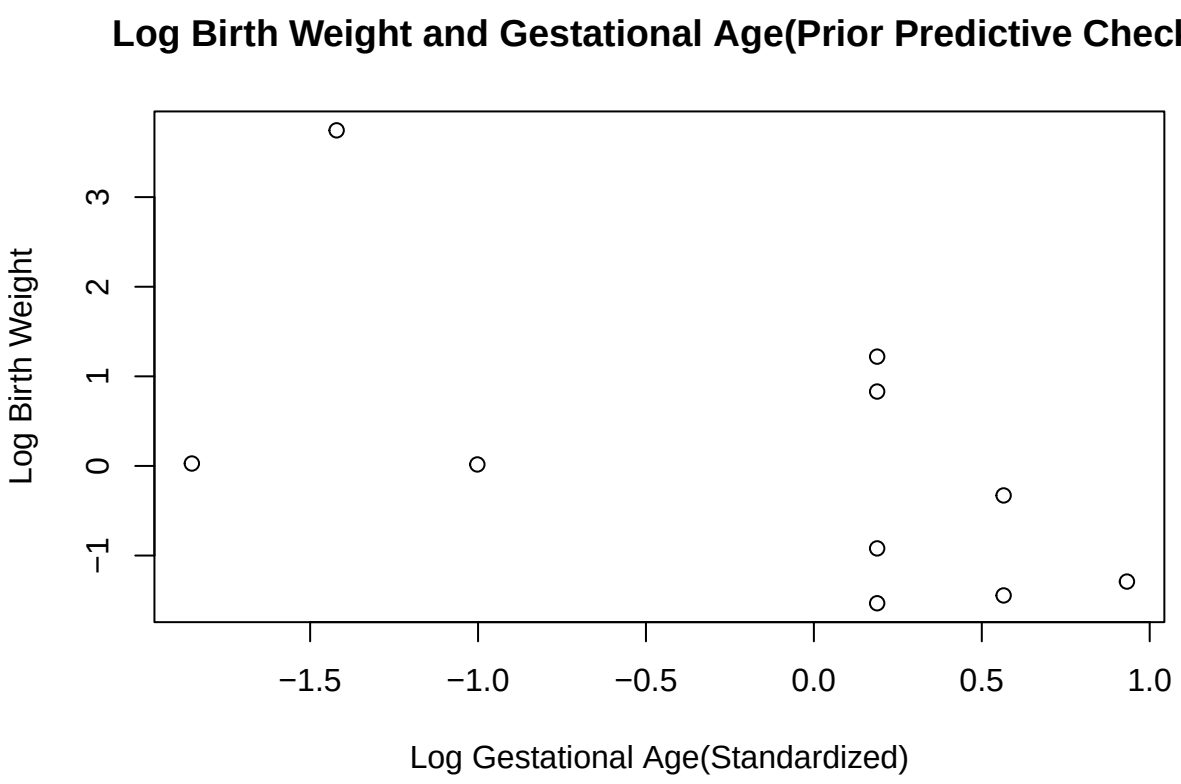
We plot the histogram of the generated birth weights.

```
set.seed(28192)
beta_1 = rnorm(1000, mean =0,sd= 1)
beta_2 = rnorm(1000,mean=0,sd=1)
sigma = abs(rnorm(1000,mean=0,sd=1))

prior_check = rnorm(1000, mean = beta_1+beta_2*scale(log(ds$gest))[1:1000], sd = sigma)

hist(prior_check, xlab = "Log(Weight)",
     main = "Prior Predictive Check", breaks = 20)
```




We plot the first ten data points.

```
plot(prior_check[1:10]~scale(log(ds$gest))[1:10],
     xlab=("Log Gestational Age(Standardized)"), ylab=("Log Birth Weight"),
     main=("Log Birth Weight and Gestational Age(Prior Predictive Check)"))
```




## 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)
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line found on 'C:
## \Users\choip\OneDrive\Documents\GitHub\STA2201H_Yeonjoon_Choi\simple_weight.stan'

##
```

```
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000996 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 9.96 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: 1.665 seconds (Warm-up)
## Chain 1:                1.379 seconds (Sampling)
## Chain 1:                3.044 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: 1.674 seconds (Warm-up)
## Chain 2:                1.493 seconds (Sampling)
## Chain 2:                3.167 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000372 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 3.72 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: 1.632 seconds (Warm-up)
## Chain 3:                1.135 seconds (Sampling)
## Chain 3:                2.767 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000355 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 3.55 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: 1.061 seconds (Warm-up)
## Chain 4:                0.902 seconds (Sampling)
## Chain 4:                1.963 seconds (Total)
## Chain 4:

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

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

```
##              mean      se_mean          sd      2.5%       25%       50%
## beta[1] 1.1624783 8.160385e-05 0.002856578 1.1570200 1.1604786 1.1625011
## beta[2] 0.1437529 8.295075e-05 0.002912236 0.1381284 0.1416970 0.1436747
## sigma   0.1690330 1.113724e-04 0.001902828 0.1652694 0.1677842 0.1690763
##               75%     97.5%     n_eff      Rhat
## beta[1] 1.1644669 1.1681028 1225.3801 0.9978044
## beta[2] 0.1456716 0.1495180 1232.5721 0.9998714
## sigma   0.1702528 0.1727953  291.9066 1.0146111
```

## Question 3

Based on model 3, give an estimate of the expected birthweight of a baby who was born at a gestational age of 37 weeks.

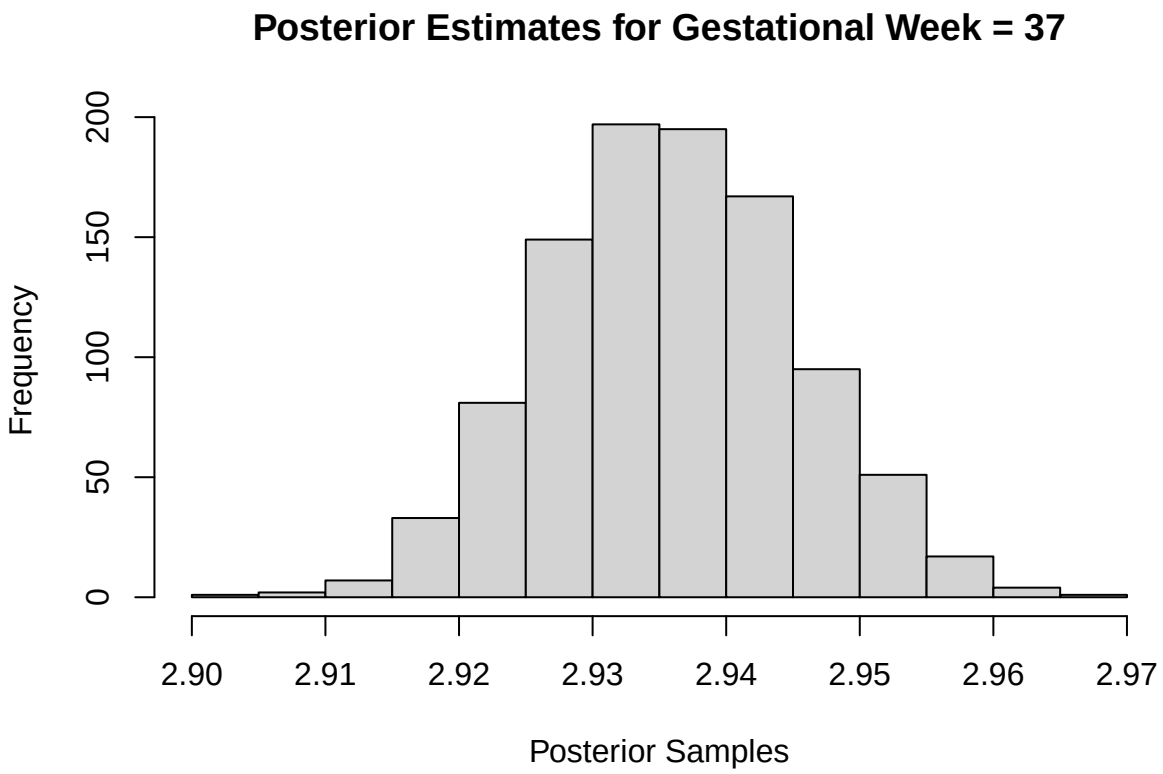
We can output the following point prediction(which is in Kg, not in log weight), using the mean of each parameter in the posterior samples.

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

```
## [1] 2.935874
```

We can also plot the posterior estimates.

```
post_samples = extract(mod1)


hist(exp(post_samples[["beta"]][,1]+((log(37)-mean(log(ds$gest)))/sd(log(ds$gest)))*post_samples[["beta
     xlab = "Posterior Samples",
     ylab = "Frequency",
     main = "Posterior Estimates for Gestational Week = 37")
```




## Question 4

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

My stan file is in outside the all the folders in my github. You will find a file named model_2_week_6.stan in my github.

```
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,
                  preterm = ifelse(ds$preterm == "Y",1,0))


my_mod2 = stan(data = stan_data,
             file = here("model_2_week_6.stan"),
             iter = 1500,
             seed = 243)
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line found on 'C:
## \Users\choip\OneDrive\Documents\GitHub\STA2201H_Yeonjoon_Choi\model_2_week_6.stan'

##
```

```
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.001056 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 10.56 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 1: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 1: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 1: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 1: Iteration:  600 / 1500 [ 40%]  (Warmup)
## Chain 1: Iteration:  750 / 1500 [ 50%]  (Warmup)
## Chain 1: Iteration:  751 / 1500 [ 50%]  (Sampling)
## Chain 1: Iteration:  900 / 1500 [ 60%]  (Sampling)
## Chain 1: Iteration: 1050 / 1500 [ 70%]  (Sampling)
## Chain 1: Iteration: 1200 / 1500 [ 80%]  (Sampling)
## Chain 1: Iteration: 1350 / 1500 [ 90%]  (Sampling)
## Chain 1: Iteration: 1500 / 1500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 8.679 seconds (Warm-up)
## Chain 1:                9.611 seconds (Sampling)
## Chain 1:                18.29 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.001583 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 15.83 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 2: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 2: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 2: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 2: Iteration:  600 / 1500 [ 40%]  (Warmup)
## Chain 2: Iteration:  750 / 1500 [ 50%]  (Warmup)
## Chain 2: Iteration:  751 / 1500 [ 50%]  (Sampling)
## Chain 2: Iteration:  900 / 1500 [ 60%]  (Sampling)
## Chain 2: Iteration: 1050 / 1500 [ 70%]  (Sampling)
## Chain 2: Iteration: 1200 / 1500 [ 80%]  (Sampling)
## Chain 2: Iteration: 1350 / 1500 [ 90%]  (Sampling)
## Chain 2: Iteration: 1500 / 1500 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 8.774 seconds (Warm-up)
## Chain 2:                9.497 seconds (Sampling)
## Chain 2:                18.271 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.0007 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 7 seconds.
```

```
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 3: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 3: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 3: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 3: Iteration:  600 / 1500 [ 40%]  (Warmup)
## Chain 3: Iteration:  750 / 1500 [ 50%]  (Warmup)
## Chain 3: Iteration:  751 / 1500 [ 50%]  (Sampling)
## Chain 3: Iteration:  900 / 1500 [ 60%]  (Sampling)
## Chain 3: Iteration: 1050 / 1500 [ 70%]  (Sampling)
## Chain 3: Iteration: 1200 / 1500 [ 80%]  (Sampling)
## Chain 3: Iteration: 1350 / 1500 [ 90%]  (Sampling)
## Chain 3: Iteration: 1500 / 1500 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 12.542 seconds (Warm-up)
## Chain 3:                8.412 seconds (Sampling)
## Chain 3:                20.954 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000625 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 6.25 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 4: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 4: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 4: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 4: Iteration:  600 / 1500 [ 40%]  (Warmup)
## Chain 4: Iteration:  750 / 1500 [ 50%]  (Warmup)
## Chain 4: Iteration:  751 / 1500 [ 50%]  (Sampling)
## Chain 4: Iteration:  900 / 1500 [ 60%]  (Sampling)
## Chain 4: Iteration: 1050 / 1500 [ 70%]  (Sampling)
## Chain 4: Iteration: 1200 / 1500 [ 80%]  (Sampling)
## Chain 4: Iteration: 1350 / 1500 [ 90%]  (Sampling)
## Chain 4: Iteration: 1500 / 1500 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 9.725 seconds (Warm-up)
## Chain 4:                10.167 seconds (Sampling)
## Chain 4:                19.892 seconds (Total)
## Chain 4:
```

## Question 5

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

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

```
##              mean      se_mean          sd       2.5%        25%       50%
## beta[1] 1.1697241 1.385590e-04 0.002742186 1.16453578 1.16767109 1.1699278
## beta[2] 0.5563133 5.835253e-03 0.058054991 0.43745504 0.51708255 0.5561553
## beta[3] 0.1020960 1.481816e-04 0.003669476 0.09459462 0.09997153 0.1020339
## beta[4] 0.1967671 1.129799e-03 0.012458398 0.17164533 0.18817091 0.1974114
## sigma   0.1610727 9.950037e-05 0.001782004 0.15784213 0.15978020 0.1610734
##               75%     97.5%     n_eff      Rhat
## beta[1] 1.1716235 1.1750167 391.67359 1.0115970
## beta[2] 0.5990427 0.6554967  98.98279 1.0088166
## beta[3] 0.1044230 0.1093843 613.22428 0.9978156
## beta[4] 0.2064079 0.2182454 121.59685 1.0056875
## sigma   0.1623019 0.1646189 320.75100 1.0104805
```

My result

```
summary(my_mod2)$summary[c("beta[1]", "beta[2]","beta[3]", "beta[4]", "sigma"),]
```

```
##              mean      se_mean          sd       2.5%        25%       50%
## beta[1] 1.1696112 4.101928e-05 0.002534302 1.16474346 1.16780607 1.1695909
## beta[2] 0.1018564 6.349936e-05 0.003495407 0.09509927 0.09939896 0.1018111
## beta[3] 0.5607959 1.641828e-03 0.064168348 0.43180913 0.51791933 0.5617302
## beta[4] 0.1980700 3.421096e-04 0.013196737 0.17197248 0.18937990 0.1984452
## sigma   0.1612285 4.266815e-05 0.001861034 0.15773258 0.15994589 0.1612178
##               75%     97.5%    n_eff      Rhat
## beta[1] 1.1713554 1.1746205 3817.163 1.0000948
## beta[2] 0.1042260 0.1087761 3030.099 0.9996405
## beta[3] 0.6042576 0.6872945 1527.518 0.9994423
## beta[4] 0.2070115 0.2246609 1487.999 0.9990718
## sigma   0.1624606 0.1650409 1902.397 1.0008281
```

The results are indeed similar.

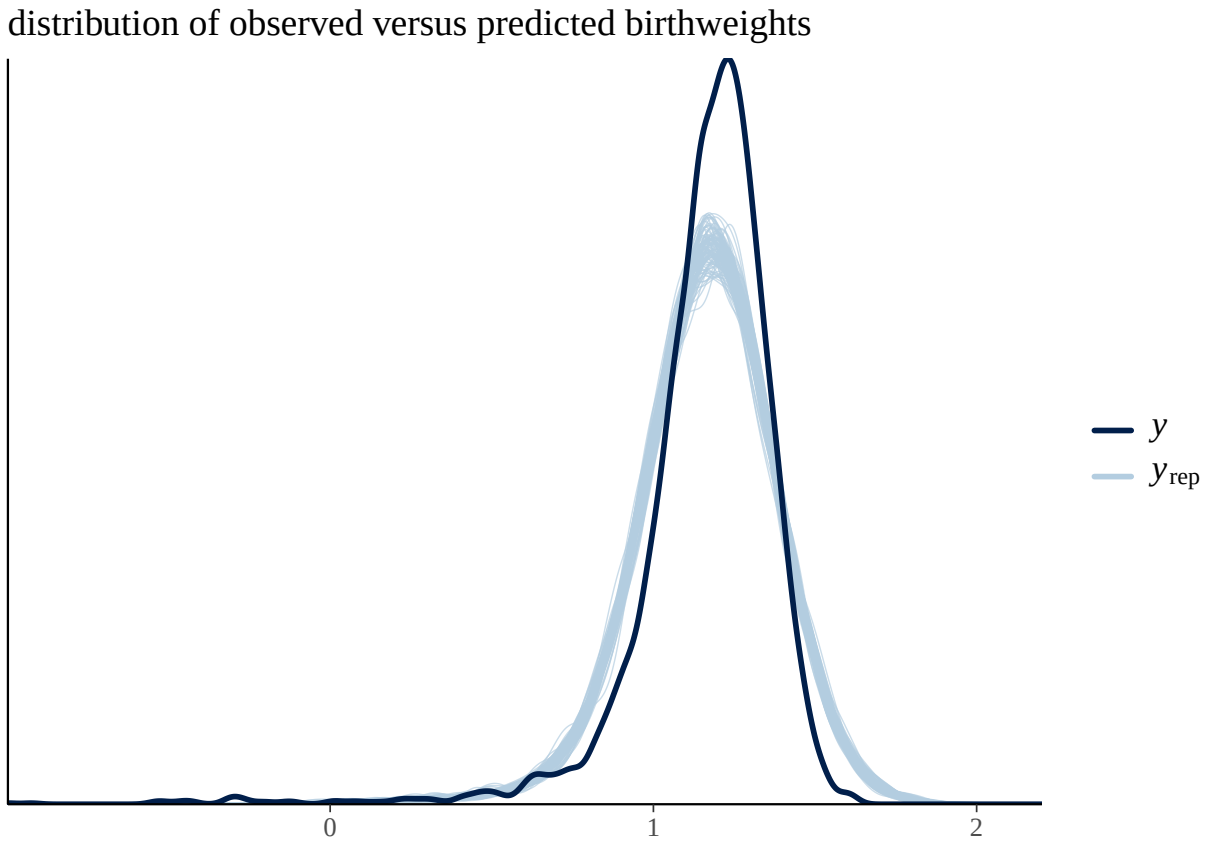
# PPCs

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

```
set.seed(1856)
y <- ds$log_weight
yrep1 <- extract(mod1)[["log_weight_rep"]]
yrep2 <- extract(mod2)[["log_weight_rep"]]
dim(yrep1)
```

```
## [1] 1000 3842
```

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




For each posterior sample, we create a data set of length 3842. We have 1000 posterior samples, so we obtain 1000 data set of of length 3842.

## Question 6

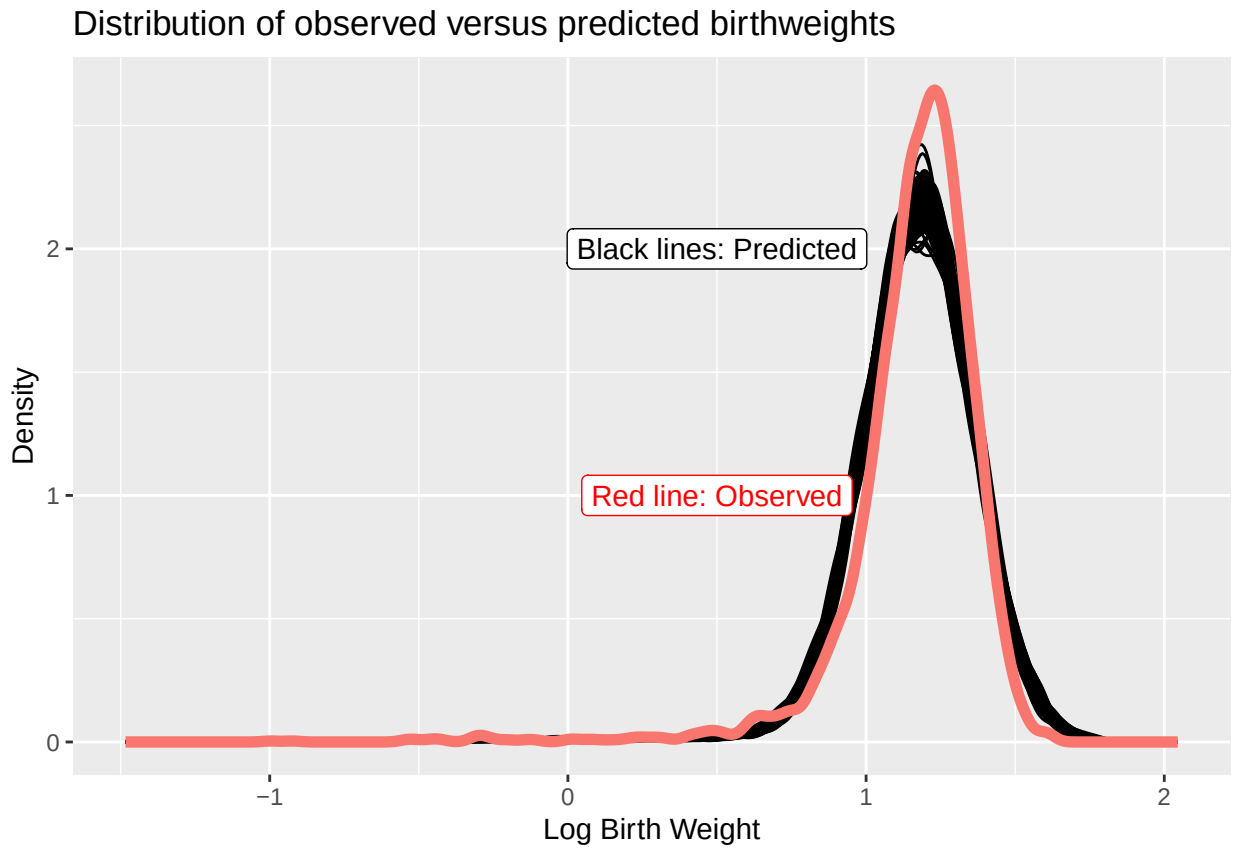
Make a similar plot to the one above but for model 2, and **not** using the bayes plot in built function (i.e. do it yourself just with `geom_density`)

```
set.seed(32222)
samp100 = sample(nrow(yrep2), 100)


p = ggplot()

for (i in 1:100){
  p = p+geom_density(aes_string(x = yrep2[samp100[i], ]))
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation ideoms with `aes()`
```

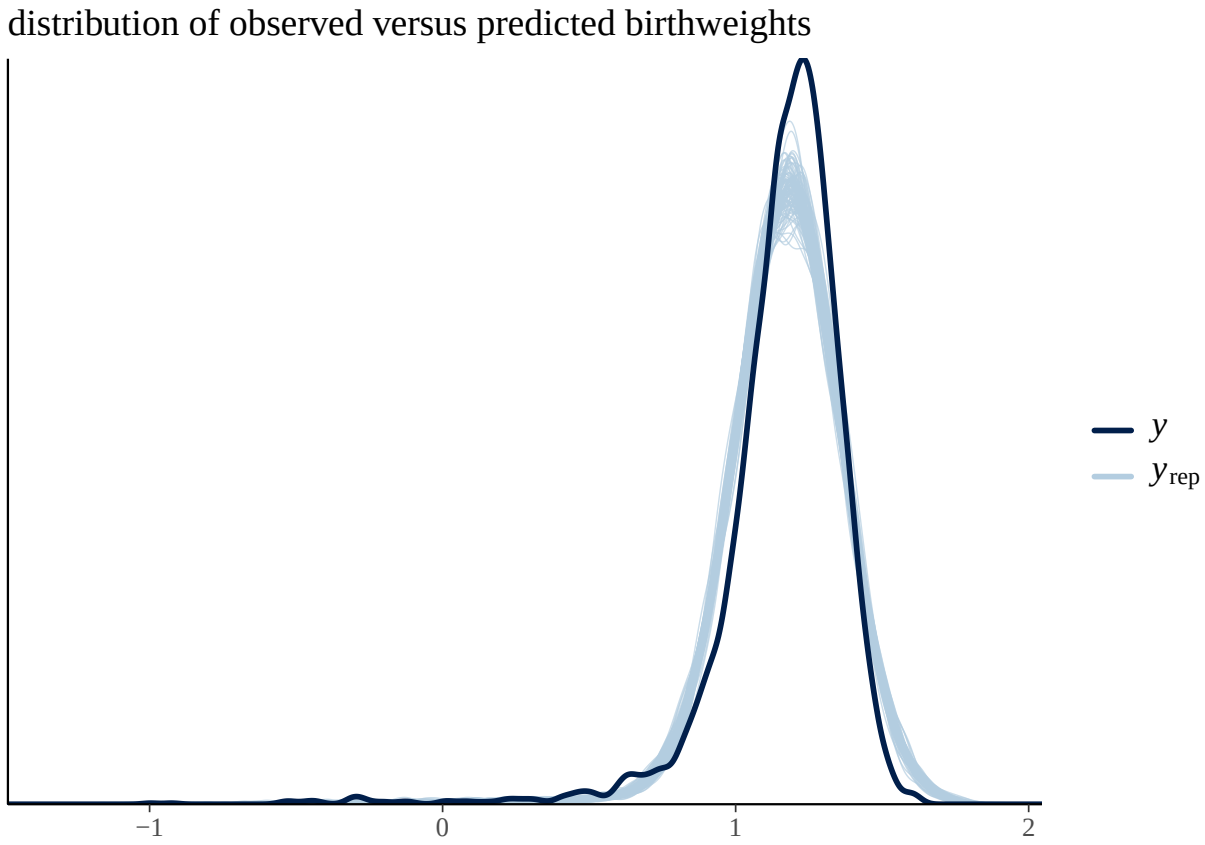
```
p = p+
  geom_density(aes(x = log_weight, color="red"), data = ds, linewidth =2)+
  theme(legend.position="none")+
  xlab("Log Birth Weight")+
```

```
  ylab("Density")+
  labs(title = "Distribution of observed versus predicted birthweights")+
  geom_label(aes(x=0.5, y=1, label="Red line: Observed"), color="red") +
  geom_label(aes(x=0.5, y=2, label="Black lines: Predicted"), color="black")



p
```




And we can compare with the output using ppc_dens_overlay

```
ppc_dens_overlay(y, yrep2[samp100, ])  + ggtitle("distribution of observed versus predicted birthweights
```



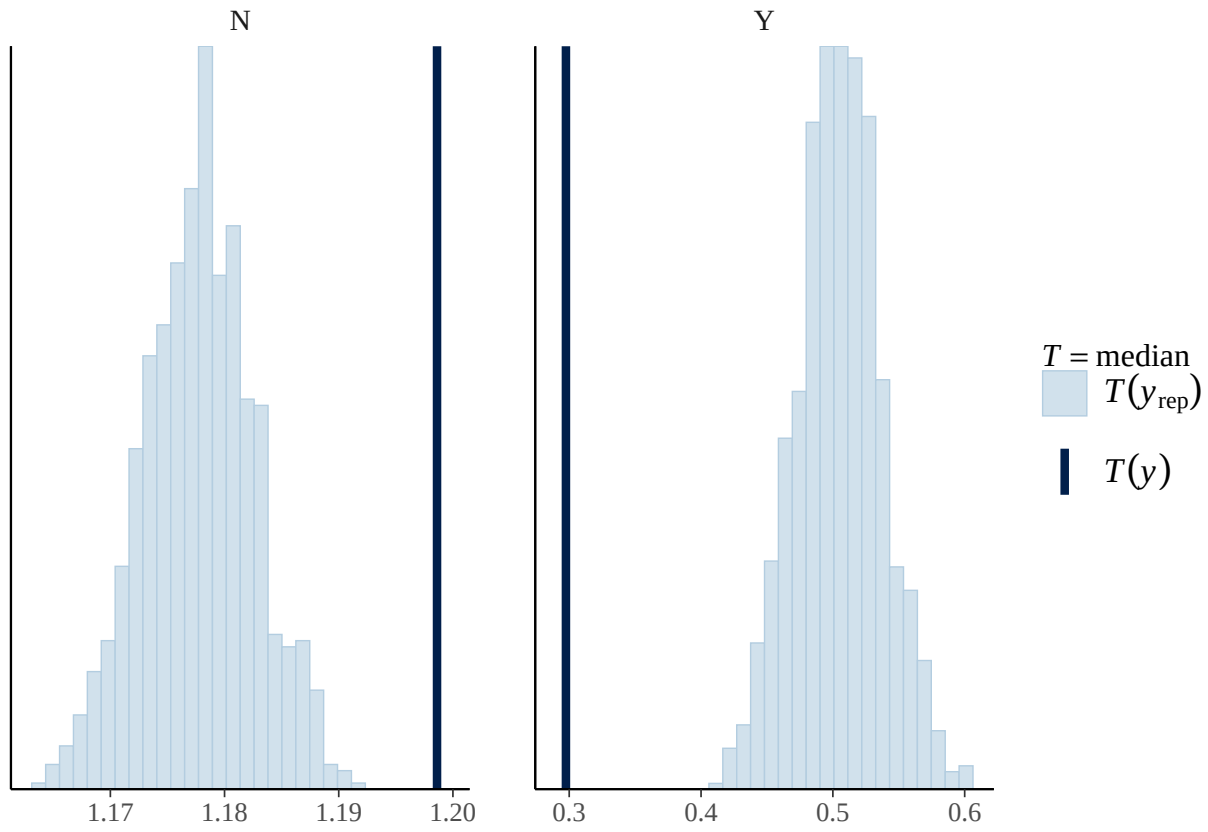

## Test statistics

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

E.g. medians by prematurity for Model 1

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

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



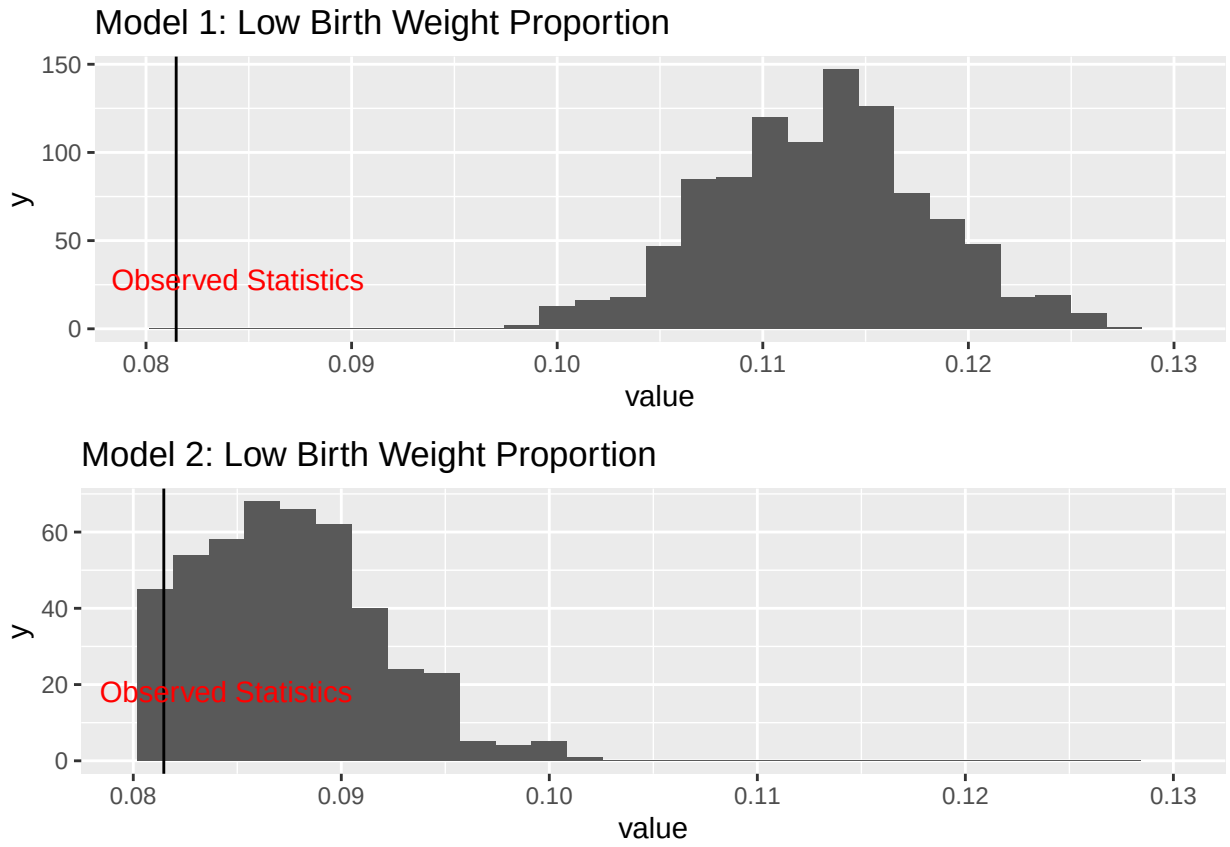

## Question 7

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).

```
test_stat = length(which(ds$low.birth =="Y"))/nrow(ds)

test_mod_1 = rep(NA,1000)

for (i in 1:1000){
  test_mod_1[i] = length(which(exp(yrep1[i, ])<2.5))/nrow(ds)
}

test_mod_2 = rep(NA,500)

for (i in 1:500){
  test_mod_2[i] = length( which(exp(yrep2[i, ])<2.5))/nrow(ds)
}


p1 = ggplot(mapping = aes(test_mod_1))+
  geom_histogram()+
  geom_vline(xintercept = test_stat)+
  xlim(c(0.08, 0.13))+
  xlab("value")+
```

```
  annotate("text", x = test_stat+0.003, y = 10, label = "Observed Statistics", vjust = -1, color = "red
  labs(title = "Model 1: Low Birth Weight Proportion")



p2 = ggplot(mapping = aes(test_mod_2))+
  geom_histogram()+
  geom_vline(xintercept = test_stat)+
  xlim(c(0.08, 0.13))+
  xlab("value")+
  annotate("text", x = test_stat+0.003, y = 10, label = "Observed Statistics", vjust = -1, color = "red
  labs(title = "Model 2: Low Birth Weight Proportion")

grid.arrange(p1,p2)
```

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

## Warning: Removed 2 rows containing missing values (`geom_bar()`).

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

## Warning: Removed 41 rows containing non-finite values (`stat_bin()`).
## Removed 2 rows containing missing values (`geom_bar()`).
```



# LOO

Finally let's calculate the LOO elpd for each model and compare. The first step of this is to get the point-wise log likelihood estimates from each model:

```
loglik1 <- extract(mod1)[["log_lik"]]
loglik2 <- extract(mod2)[["log_lik"]]
```

And then we can use these in the `loo` function to get estimates for the elpd. Note the `save_psis = TRUE` argument saves the calculation for each simulated draw, which is needed for the LOO-PIT calculation below.

```
loo1 <- loo(loglik1, save_psis = TRUE)
```

```
## 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.
```

```
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.
```

Look at the output:

```
loo1
```

```
##
## Computed from 1000 by 3842 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo   1377.0  72.4
## p_loo         9.8   1.4
## looic     -2754.1 144.8
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
loo2
```

```
##
## Computed from 500 by 3842 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo   1552.8  70.0
## p_loo        14.8   2.3
## looic     -3105.6 139.9
## ------
## Monte Carlo SE of elpd_loo is 0.2.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

Comparing the two models tells us Model 2 is better:

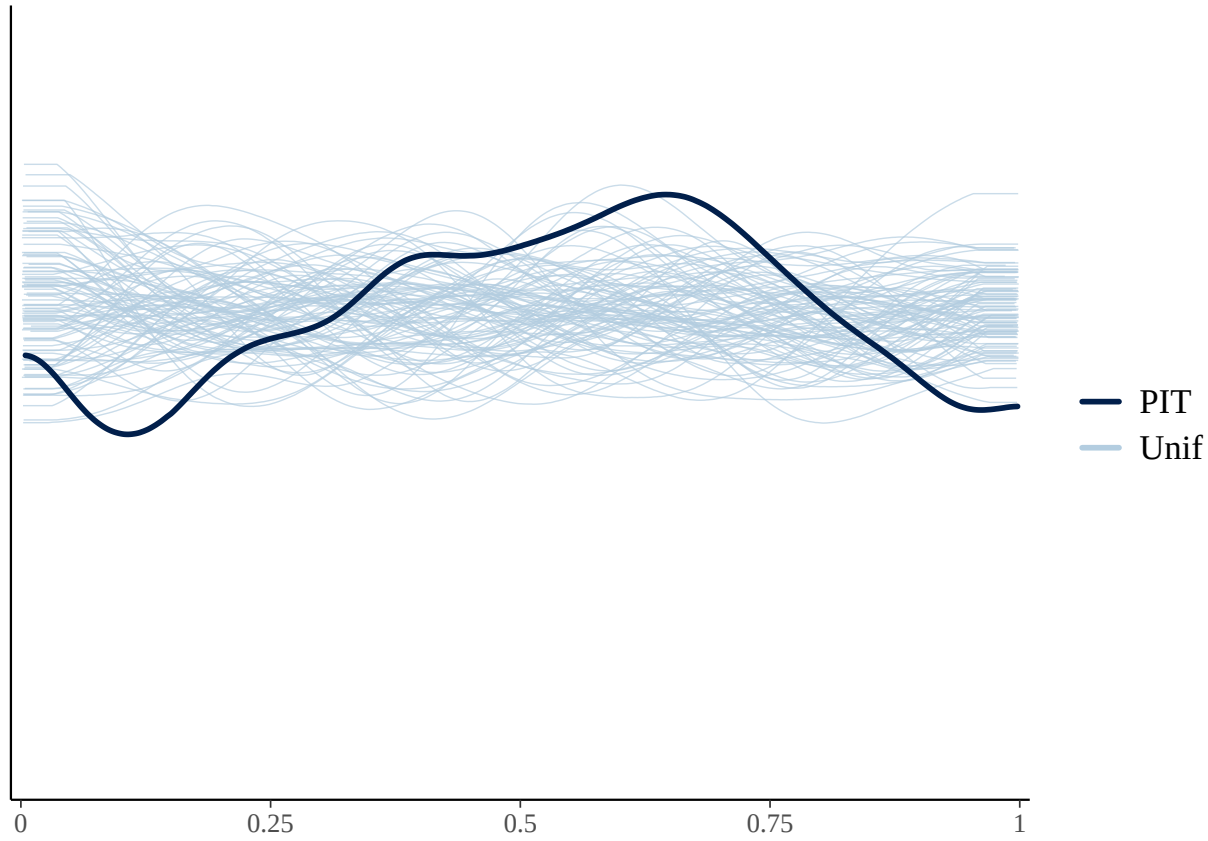
```
loo_compare(loo1, loo2)
```

```
##        elpd_diff se_diff
## model2    0.0       0.0
## model1 -175.8      36.2
```

We can also compare the LOO-PIT of each of the models to standard uniforms. The both do pretty well.

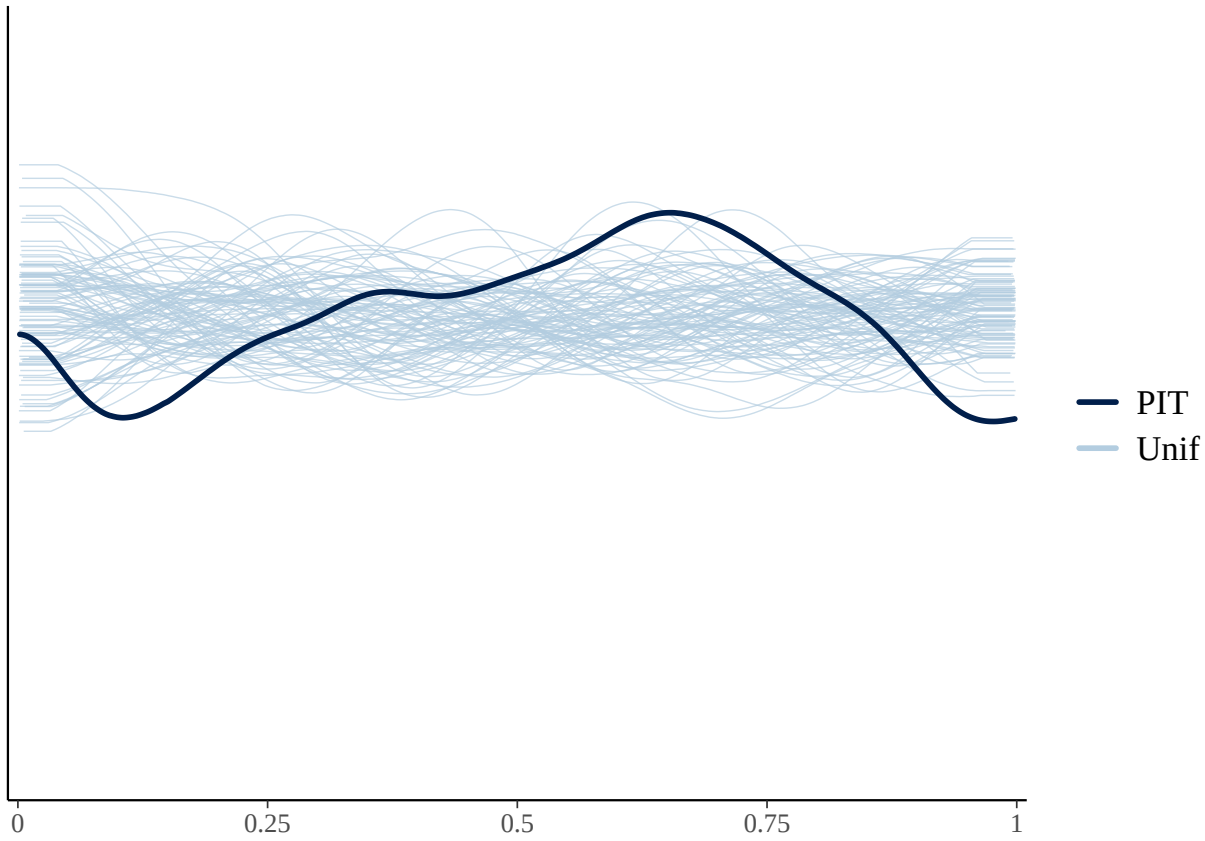
```
ppc_loo_pit_overlay(yrep = yrep1, y = y, lw = weights(loo1$psis_object))
```

```
## NOTE: The kernel density estimate assumes continuous observations and is not optimal for discrete obs
```




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

```
## NOTE: The kernel density estimate assumes continuous observations and is not optimal for discrete obs
```




### Bonus question (not required)

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

### Question 8

Based on the original dataset, choose one (or more) additional covariates to add to the linear regression model. Run the model in Stan, and compare with Model 2 above on at least 2 posterior predictive checks.

We added a covariate for whether or not mother had BMI over 30, and a covariate for whether or not the mother was black.

```
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,
                  preterm = ifelse(ds$preterm == "Y",1,0),
                  black = ifelse(ds$Black == "Y",1,0),
                  obesity = ifelse(ds$Obesity == "Y",1,0))


my_mod3 = stan(data = stan_data,
```

```
            file = here("model_3_week_6.stan"),
            iter = 1500,
            seed = 243)
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line found on 'C:
## \Users\choip\OneDrive\Documents\GitHub\STA2201H_Yeonjoon_Choi\model_3_week_6.stan'
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.00444 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 44.4 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 1: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 1: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 1: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 1: Iteration:  600 / 1500 [ 40%]  (Warmup)
## Chain 1: Iteration:  750 / 1500 [ 50%]  (Warmup)
## Chain 1: Iteration:  751 / 1500 [ 50%]  (Sampling)
## Chain 1: Iteration:  900 / 1500 [ 60%]  (Sampling)
## Chain 1: Iteration: 1050 / 1500 [ 70%]  (Sampling)
## Chain 1: Iteration: 1200 / 1500 [ 80%]  (Sampling)
## Chain 1: Iteration: 1350 / 1500 [ 90%]  (Sampling)
## Chain 1: Iteration: 1500 / 1500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 22.985 seconds (Warm-up)
## Chain 1:                23.344 seconds (Sampling)
## Chain 1:                46.329 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.002358 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 23.58 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 2: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 2: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 2: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 2: Iteration:  600 / 1500 [ 40%]  (Warmup)
## Chain 2: Iteration:  750 / 1500 [ 50%]  (Warmup)
## Chain 2: Iteration:  751 / 1500 [ 50%]  (Sampling)
## Chain 2: Iteration:  900 / 1500 [ 60%]  (Sampling)
## Chain 2: Iteration: 1050 / 1500 [ 70%]  (Sampling)
## Chain 2: Iteration: 1200 / 1500 [ 80%]  (Sampling)
## Chain 2: Iteration: 1350 / 1500 [ 90%]  (Sampling)
## Chain 2: Iteration: 1500 / 1500 [100%]  (Sampling)
## Chain 2:
```

```
## Chain 2:  Elapsed Time: 22.609 seconds (Warm-up)
## Chain 2:                24.038 seconds (Sampling)
## Chain 2:                46.647 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.00138 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 13.8 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 3: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 3: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 3: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 3: Iteration:  600 / 1500 [ 40%]  (Warmup)
## Chain 3: Iteration:  750 / 1500 [ 50%]  (Warmup)
## Chain 3: Iteration:  751 / 1500 [ 50%]  (Sampling)
## Chain 3: Iteration:  900 / 1500 [ 60%]  (Sampling)
## Chain 3: Iteration: 1050 / 1500 [ 70%]  (Sampling)
## Chain 3: Iteration: 1200 / 1500 [ 80%]  (Sampling)
## Chain 3: Iteration: 1350 / 1500 [ 90%]  (Sampling)
## Chain 3: Iteration: 1500 / 1500 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 27.905 seconds (Warm-up)
## Chain 3:                37.178 seconds (Sampling)
## Chain 3:                65.083 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.002366 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 23.66 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 4: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 4: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 4: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 4: Iteration:  600 / 1500 [ 40%]  (Warmup)
## Chain 4: Iteration:  750 / 1500 [ 50%]  (Warmup)
## Chain 4: Iteration:  751 / 1500 [ 50%]  (Sampling)
## Chain 4: Iteration:  900 / 1500 [ 60%]  (Sampling)
## Chain 4: Iteration: 1050 / 1500 [ 70%]  (Sampling)
## Chain 4: Iteration: 1200 / 1500 [ 80%]  (Sampling)
## Chain 4: Iteration: 1350 / 1500 [ 90%]  (Sampling)
## Chain 4: Iteration: 1500 / 1500 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 28.624 seconds (Warm-up)
## Chain 4:                34.014 seconds (Sampling)
## Chain 4:                62.638 seconds (Total)
## Chain 4:
```

Here is the fit.

```
summary(my_mod3)$summary[c("beta[1]", "beta[2]","beta[3]", "beta[4]", "beta[5]", "beta[6]", "sigma"),]
```

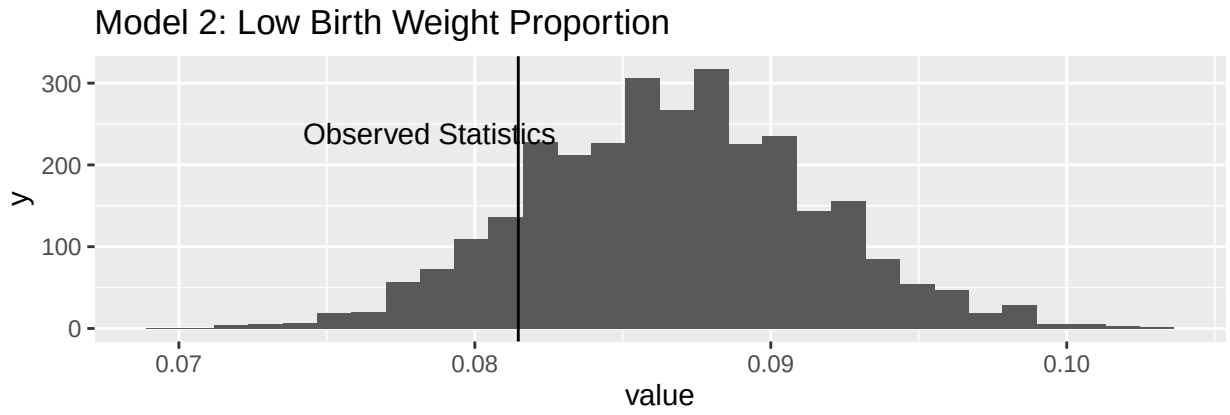
```
##                mean      se_mean          sd        2.5%         25%
## beta[1]  1.16857003 5.876925e-05 0.003284187  1.16217984  1.16630968
## beta[2]  0.10026563 5.716952e-05 0.003665074  0.09298002  0.09784059
## beta[3]  0.57216757 1.586271e-03 0.063546710  0.44665124  0.52893958
## beta[4]  0.20098006 3.230303e-04 0.013094595  0.17505543  0.19214768
## beta[5] -0.05006210 1.383814e-04 0.007221119 -0.06408144 -0.05497090
## beta[6]  0.02798574 1.010528e-04 0.005643974  0.01688066  0.02423547
## sigma    0.15985800 3.317049e-05 0.001810523  0.15644173  0.15864559
##                 50%         75%       97.5%    n_eff      Rhat
## beta[1]  1.16854048  1.17087031  1.17489245 3122.882 0.9993093
## beta[2]  0.10026276  0.10277325  0.10736647 4109.948 0.9992796
## beta[3]  0.57154387  0.61796477  0.69215306 1604.839 1.0003686
## beta[4]  0.20103467  0.21021226  0.22637082 1643.227 0.9996804
## beta[5] -0.05007737 -0.04510482 -0.03560442 2723.035 1.0006923
## beta[6]  0.02804458  0.03172030  0.03911889 3119.419 0.9992619
## sigma    0.15987433  0.16106817  0.16347926 2979.232 0.9997758
```

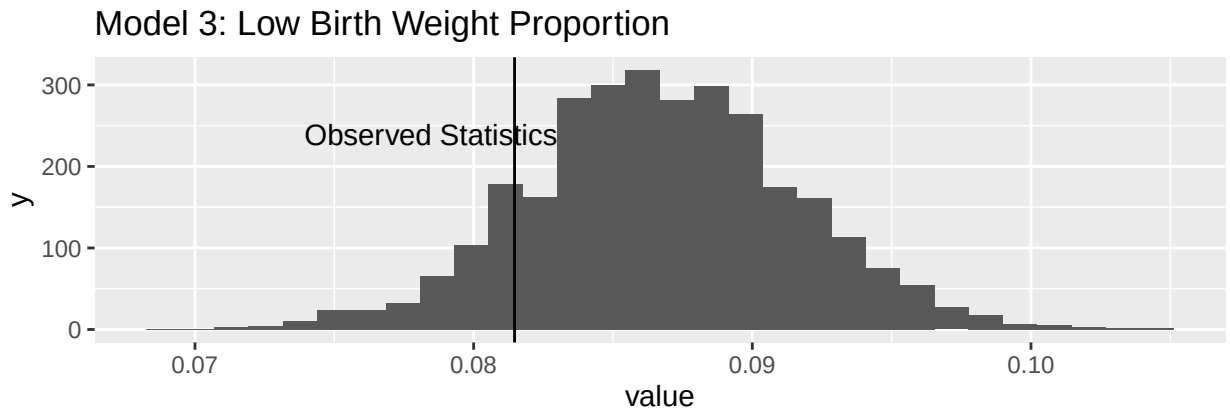
We do a posterior predictive check with proportion of low birth weight and median.

```
yrep2 = extract(my_mod2)[["log_weight_rep"]]
yrep3 = extract(my_mod3)[["log_weight_rep"]]



test_stat = length(which(ds$low.birth =="Y"))/nrow(ds)

test_mod_2 = rep(NA,3000)

for (i in 1:3000){
  test_mod_2[i] = length(which(exp(yrep2[i, ])<2.5))/nrow(ds)
}

test_mod_3 = rep(NA,3000)

for (i in 1:3000){
  test_mod_3[i] = length( which(exp(yrep3[i, ])<2.5))/nrow(ds)
}


p1 = ggplot(mapping = aes(test_mod_2))+
  geom_histogram()+
  geom_vline(xintercept = test_stat)+
  xlab("value")+
  annotate("text", x = test_stat-0.003, y = 200, label = "Observed Statistics", vjust = -1)+
  labs(title = "Model 2: Low Birth Weight Proportion")



p2 = ggplot(mapping = aes(test_mod_3))+
```

```
  geom_histogram()+
  geom_vline(xintercept = test_stat)+
  xlab("value")+
  annotate("text", x = test_stat-0.003, y = 200, label = "Observed Statistics", vjust = -1)+
  labs(title = "Model 3: Low Birth Weight Proportion")

grid.arrange(p1,p2)
```

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

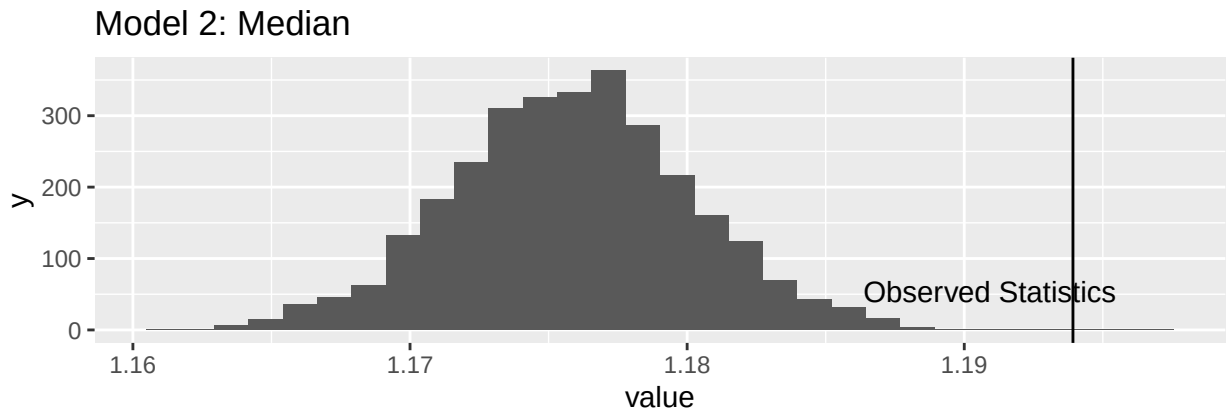


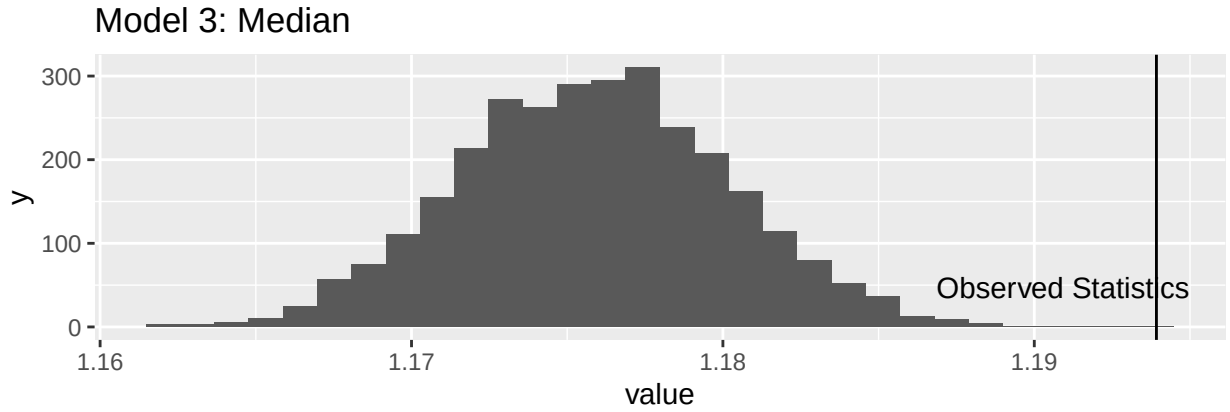




```
yrep2 = extract(my_mod2)[["log_weight_rep"]]
yrep3 = extract(my_mod3)[["log_weight_rep"]]



test_stat = log(median(ds$birthweight))

test_mod_2 = rep(NA,3000)

for (i in 1:3000){
  test_mod_2[i] = median(yrep2[i, ])
}
```

```r
test_mod_3 = rep(NA,3000)

for (i in 1:3000){
  test_mod_3[i] = median(yrep3[i, ])
}


p1 = ggplot(mapping = aes(test_mod_2))+
  geom_histogram()+
  geom_vline(xintercept = test_stat)+
  xlab("value")+
  annotate("text", x = test_stat-0.003, y = 10, label = "Observed Statistics", vjust = -1)+
  labs(title = "Model 2: Median")



p2 = ggplot(mapping = aes(test_mod_3))+
  geom_histogram()+
  geom_vline(xintercept = test_stat)+
  xlab("value")+
  annotate("text", x = test_stat-0.003, y = 10, label = "Observed Statistics", vjust = -1)+
  labs(title = "Model 3: Median")

grid.arrange(p1,p2)
```

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




And we compare loo elpd.

```
loglik2 <- extract(my_mod2)[["log_lik"]]
loglik3 <- extract(my_mod3)[["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.
```

```
loo3 <- loo(loglik3, 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.
```

```
loo_compare(loo2, loo3)
```

```
##        elpd_diff se_diff
## model2   0.0       0.0
## model1 -31.4       8.9
```

Looking at posterior predictive check, we do not see any big visual difference. However, from comparing loo elpd, model 3, with two new covariates, performed better than model 2. However, we see that difference

is smaller between model 2 and model 3, compared to the difference in model 1 and model 2. Hence, the improvmenet in performance may only have been marginal.