Week 6: Visualizing the Bayesian Workflow

Hainan Xu

19/02/24

Introduction

This lab will be looking at trying to replicate some of the visualizations in the lecture notes, involving prior and posterior predictive checks, and LOO model comparisons.

The dataset is a 0.1% of all births in the US in 2017. I've pulled out a few different variables, but as in the lecture, we'll just focus on birth weight and gestational age.

The data

Read it in, along with all our packages.

```
library(tidyverse)
library(here)
# for bayes stuff
library(rstan)
library(bayesplot)
library(loo)
library(tidybayes)

#ds <- read_rds(here("data","labs/births_2017_sample.RDS"))

ds <- readRDS("~/Documents/2024/STA2201-Methods-of-Applied-Statistics/labs/births_2017_sam
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.

Figure 1: Weight vs Gestational Age

The figure below is a scatterplot that depicts the relationships between the log in gestational age and the log in weights caterorized by whether or not the baby is pretermed or not. We observe that for pretermed baby, there are a stronger positive association between a birth-weight and gestational age; whereas for not pretermed baby, the associations are weaker. This figure also shows that pretermed babies tend to have smaller birth weights compared with not pretermed ones.

```
ds <- ds %>%
  mutate(preterm = ifelse(log(gest) <= 3.45, "Y", "N"))

ggplot(ds, aes(x = log(gest), y = log(birthweight))) +
  geom_point() +
  geom_smooth(aes(group = preterm, color = preterm), method = "lm", se = TRUE)+
  scale_color_manual(values = c("Y" = "blue", "N" = "red")) +
  theme_bw() +labs(x="log of gestational age",y="log of birth weight")
```

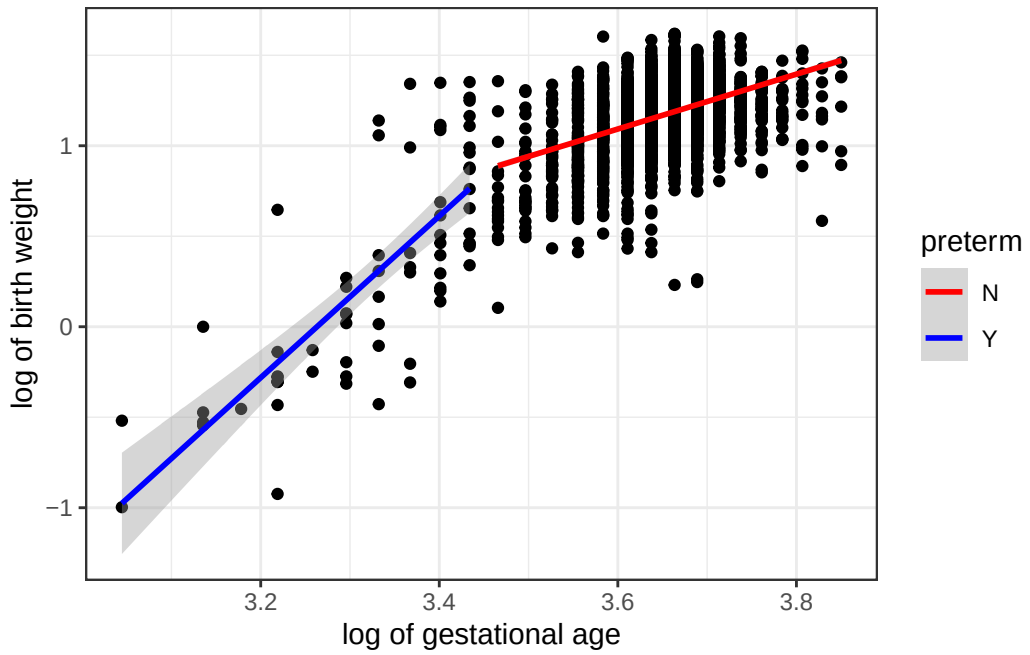
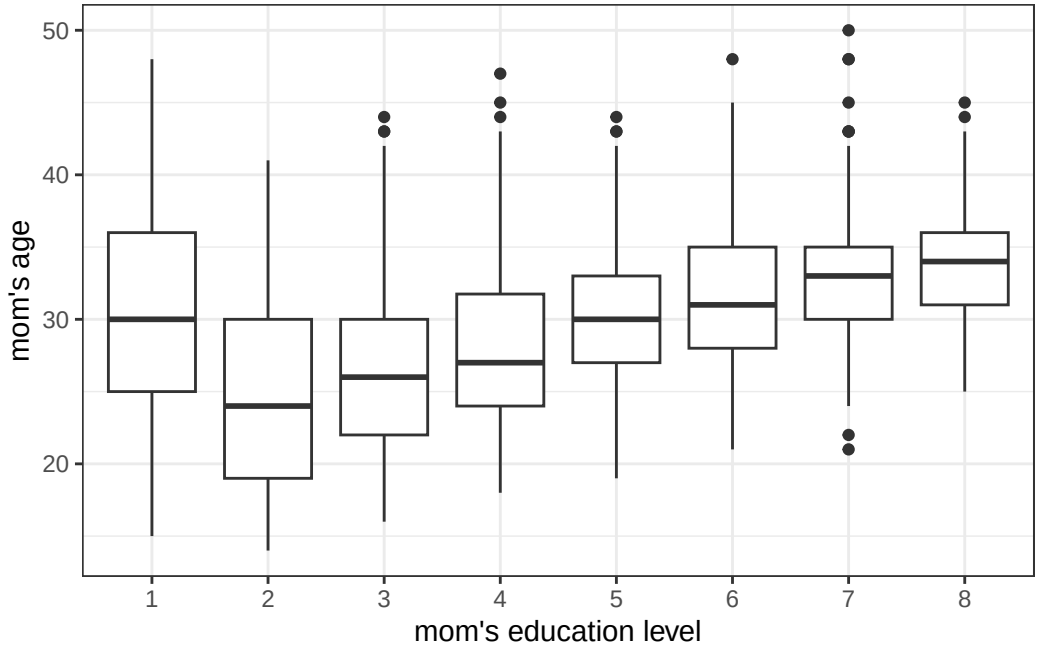


Figure 2: Mom's age vs Mom's Education

The figure below illustrate the relationships between mom's age and number of year's of mom's education. Except for the moms with 8^th grade or less, we observed a positive relationships

between the mean of mom's age and the level of educatiion received by the moms. in addition, the range of mom's age get's smaller as the education level goes up.

```
ggplot(data=ds|>filter(meduc!="9"),aes(x =as.factor(meduc),y=mager))+geom_boxplot()+theme_
```



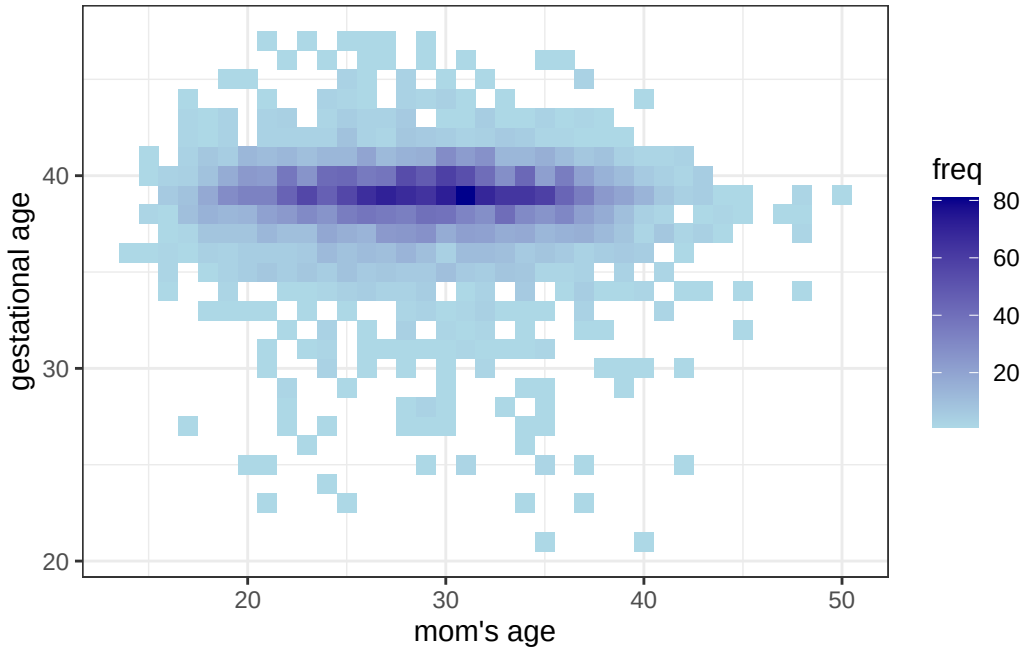

```
#observations with `meduc` == 9 (unkown) are filtered out
```

Figure 3: Mom's age and gestational age

The figure below shows the relationship between age of the mom and their gestational age. The color of each tile represent the frequency of correspongding values of age of mom and their gestational age. We observe that most womens gestational age is around 38 and 39 weeks. It seems that age of mom has no effect on the gestational age.

```
#relationship between preterm and
ds_summary <- ds %>%
  group_by(mager, gest) %>%
  summarise(freq = n(), .groups = 'drop')

ggplot(ds_summary, aes(x = mager, y = gest, fill = freq)) +
  geom_tile() +
```

```
scale_fill_gradient(low = "lightblue", high = "darkblue") +
theme_bw()   +labs(x="mom's age",y="gestational age")
```



The model

As in lecture, we will look at two candidate models

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

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

Model 2 has an interaction term between gestation and prematurity

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

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

Prior predictive checks

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

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

and for σ

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

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

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

Question 2

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

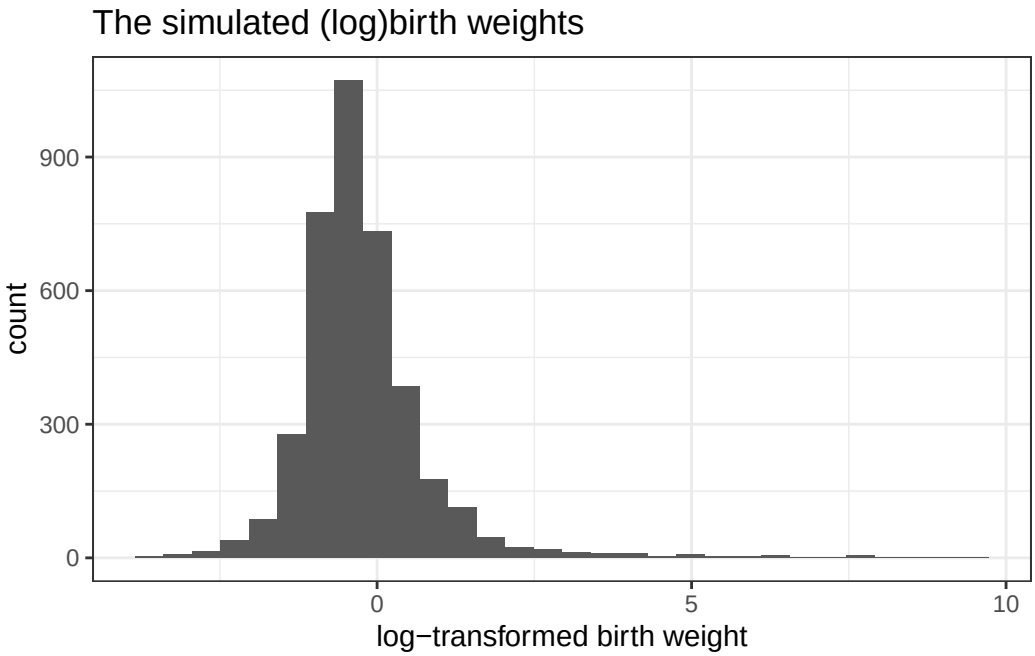
- Plot the resulting distribution of simulated (log) birth weights.
- Plot ten simulations of (log) birthweights against gestational age.

One of the resulting distribution is shown below for simulated log birth weights.

```
set.seed(123)
b1=rnorm(1000,mean=0,sd=1) |>matrix(ncol = 1)
b2=rnorm(1000,mean=0,sd=1) |>matrix(ncol = 1)

sigma=abs(rnorm(1000,0,1)) |>matrix(ncol = 1)#generate 1 sigma or many sigmas

ds2 <- ds# |> sample_n(1000)
log_gest=log(ds2$gest)
gest.stadardized=matrix((log_gest-mean(log_gest))/sd(log_gest),ncol = 1)

e<-NULL
birthweight_list <- vector("list", 1000)
for (i in 1:1000){
ei=rnorm(length(gest.stadardized),0,sigma[i])
```

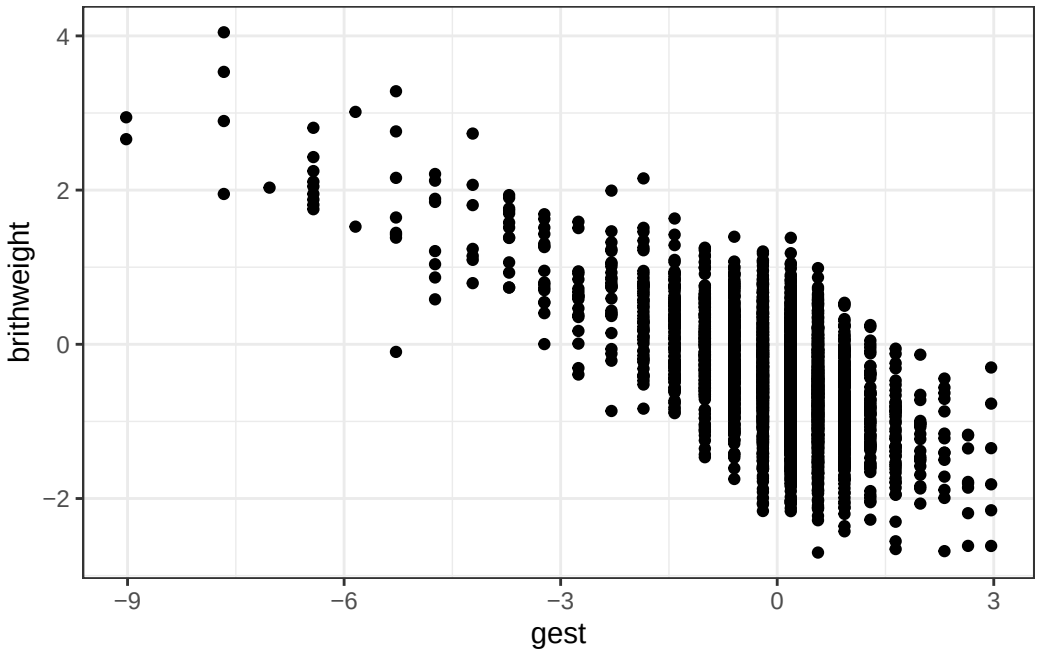
```
e=c(e,ei) |> matrix(ncol = 1)
birthweighti=b1[i]+b2[i]*gest.stadardized+ei
birthweight_list[[i]] <- birthweighti
}
```

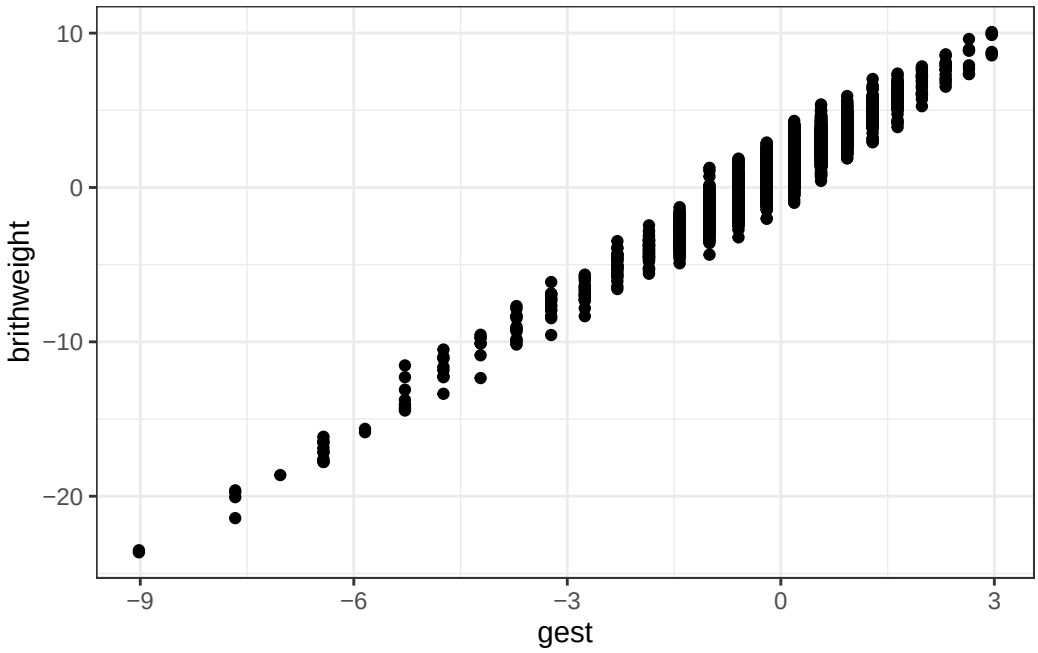
```
ggplot()+geom_histogram(aes(x=birthweight_list[[2]]))+theme_bw() + labs(x="log-transformed
```

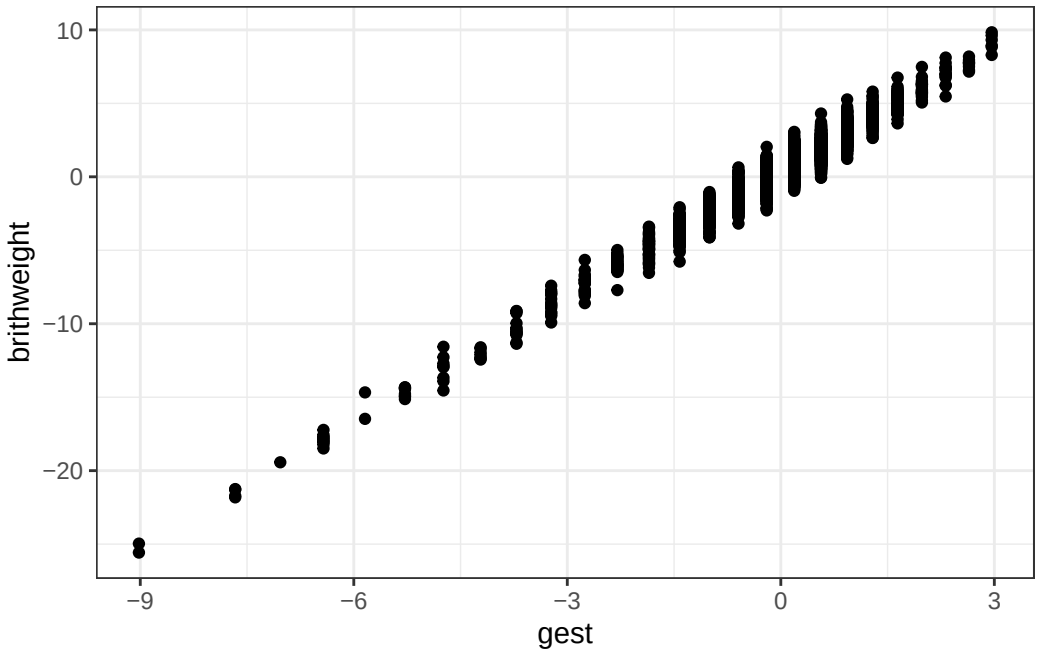


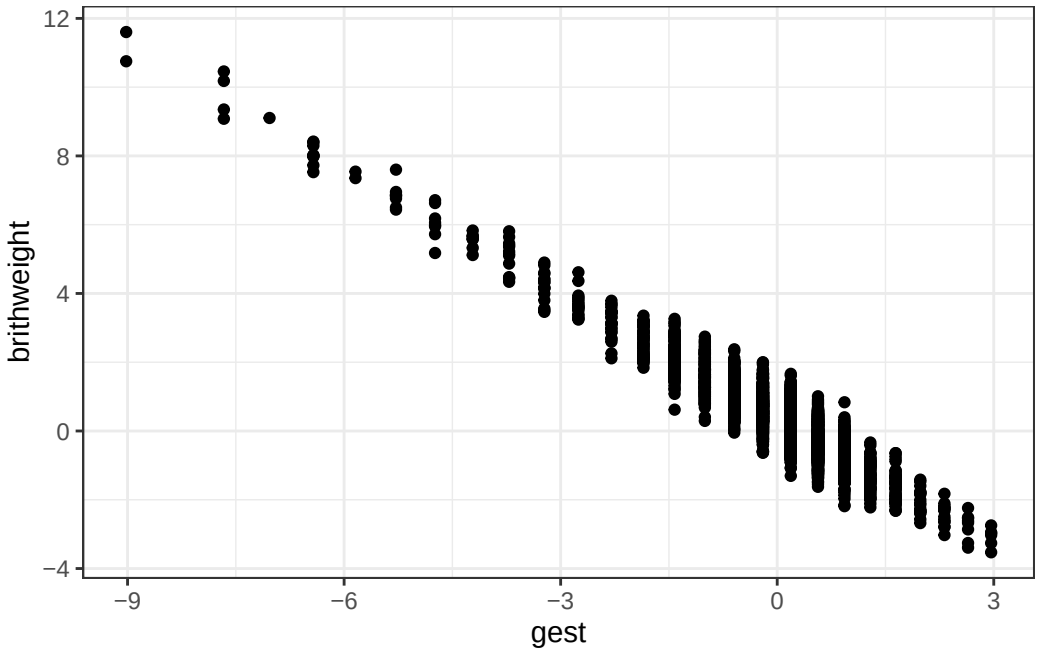
Below are 10 simulations of log birth weight against gestational age.

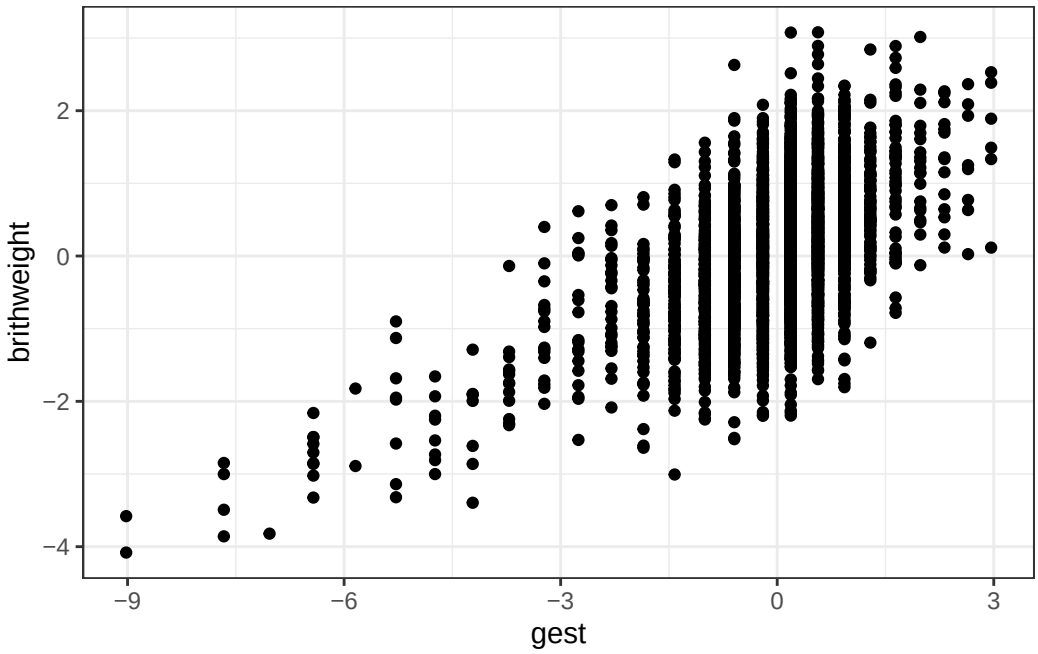
```
par(mfrow = c(4, 3))
p<- list()
for (i in 10:20){
pi<-ggplot()+geom_point(aes(x=gest.stadardized,y=birthweight_list[[i]]))+theme_bw()+labs(x
print(pi)
}
```

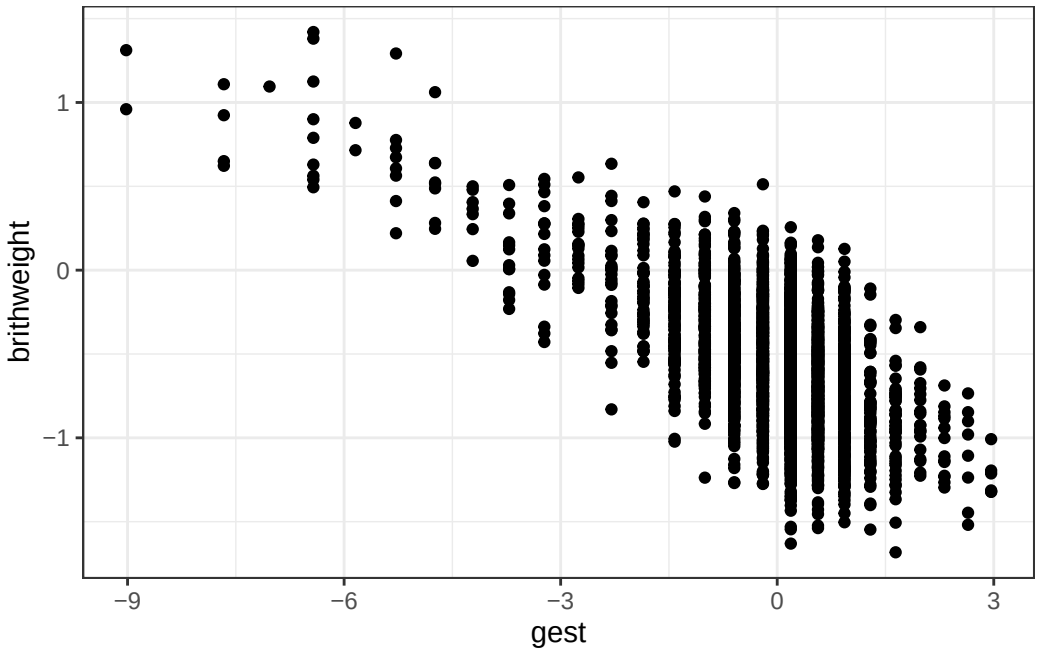

4
2
0
−2
brithweight
−9
−6
−3
0
3
gest

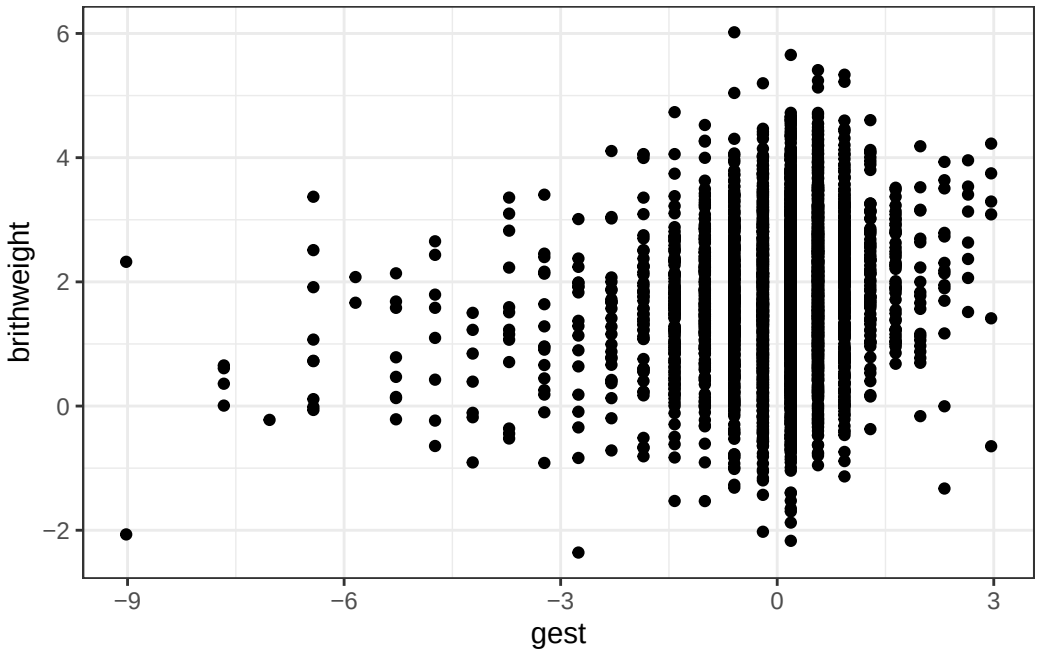

10
0
−10
−20
brithweight
−9
−6
−3
0
3
gest

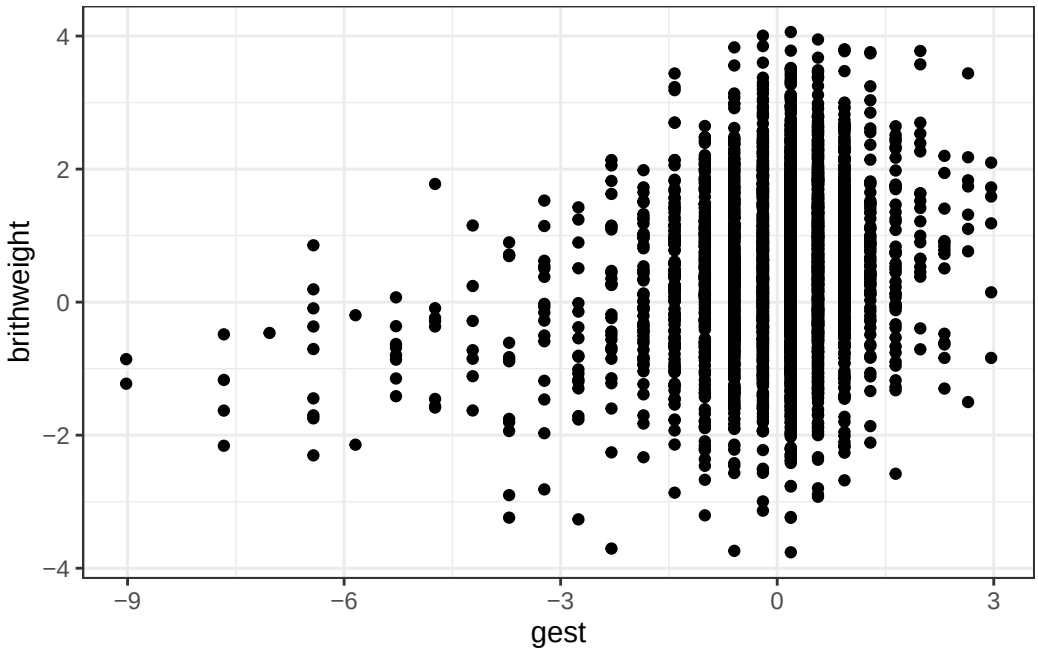

10
0
−10
−20
brithweight
−9
−6
−3
0
3
gest

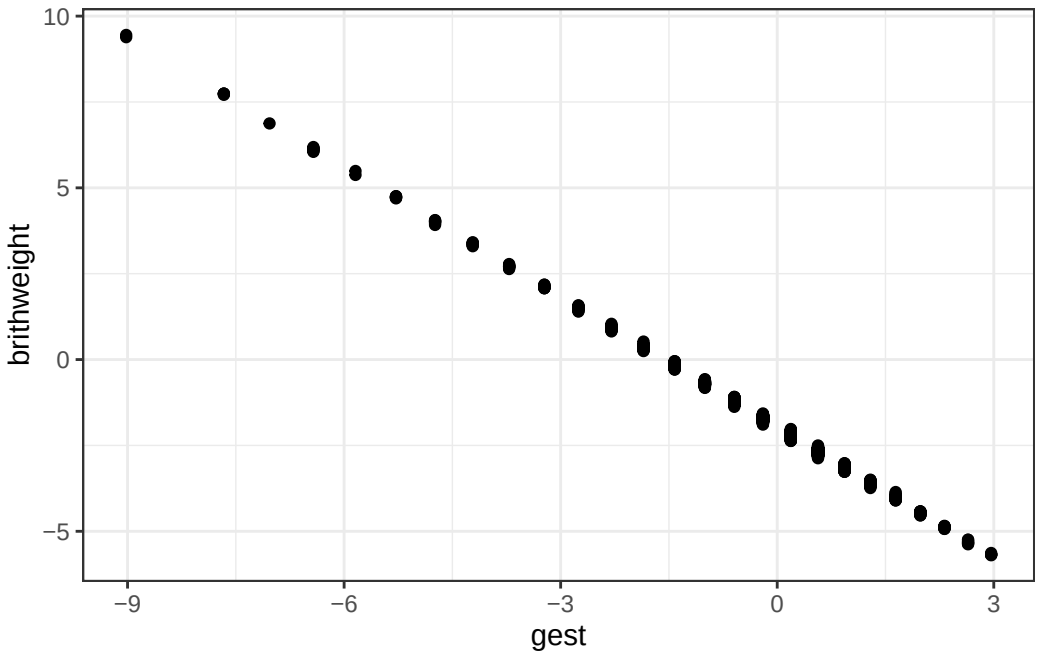

12
8
4
0
−4
brithweight
−9
−6
−3
0
3
gest

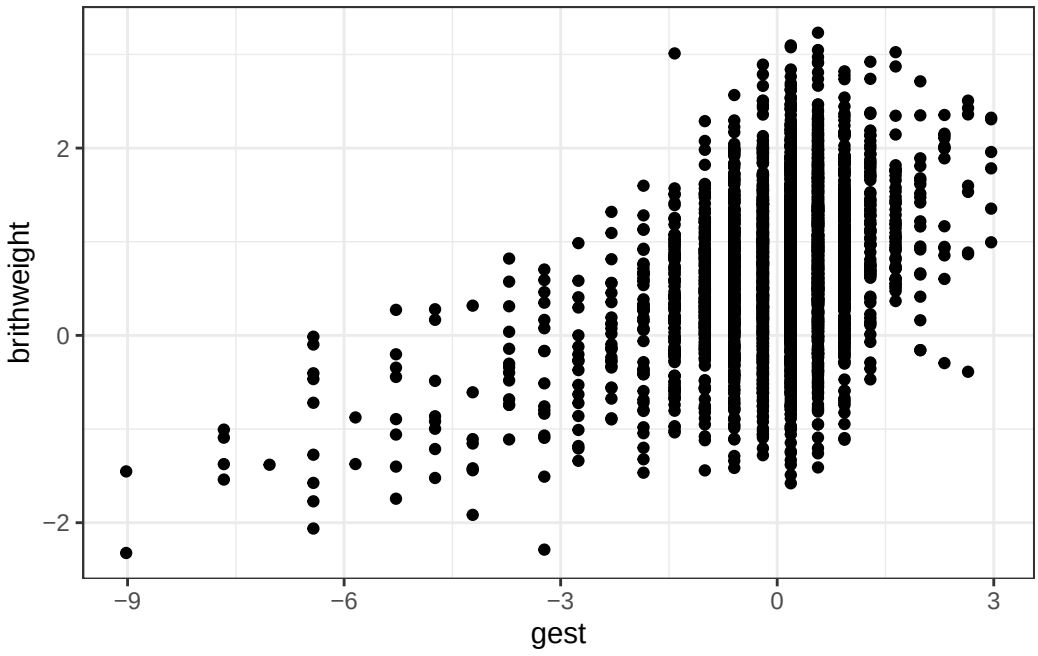

brithweight
2
0
−2
−4
−9
−6
−3
0
3
gest

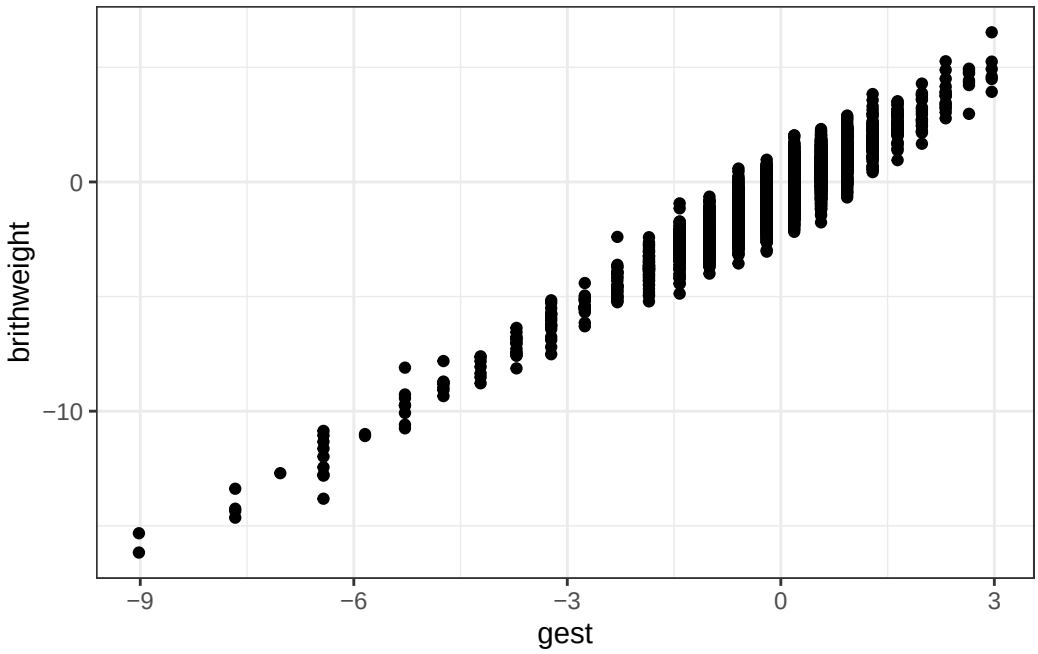

brithweight
1
0
−1
−9
−6
−3
0
3
gest


6
4
2
0
−2
brithweight
−9
−6
−3
0
3
gest


4
2
0
−2
−4
brithweight
−9
−6
−3
0
3
gest


10
5
0
−5
brithweight
−9
−6
−3
0
3
gest


2
0
−2
brithweight
−9
−6
−3
0
3
gest



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

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
```

```
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEI
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHeac
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEige
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEige
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Co
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Co
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHeac
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEige
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/Core:9
#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 0.000409 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 4.09 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: 0.371 seconds (Warm-up)
Chain 1:                0.328 seconds (Sampling)
```

```
Chain 1:                0.699 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000419 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 4.19 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: 0.36 seconds (Warm-up)
Chain 2:                0.332 seconds (Sampling)
Chain 2:                0.692 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000224 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.24 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: 0.383 seconds (Warm-up)
Chain 3:                0.295 seconds (Sampling)
Chain 3:                0.678 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000306 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 3.06 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: 0.371 seconds (Warm-up)
Chain 4:                0.296 seconds (Sampling)
Chain 4:                0.667 seconds (Total)
Chain 4:
```

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

```
             mean      se_mean          sd      2.5%       25%       50%
beta[1] 1.1625308 8.937069e-05 0.002939494 1.1567671 1.1604564 1.1625497
beta[2] 0.1437732 8.218239e-05 0.002767627 0.1385299 0.1417673 0.1437258
sigma   0.1689611 1.007199e-04 0.001848675 0.1653767 0.1677087 0.1691049
```

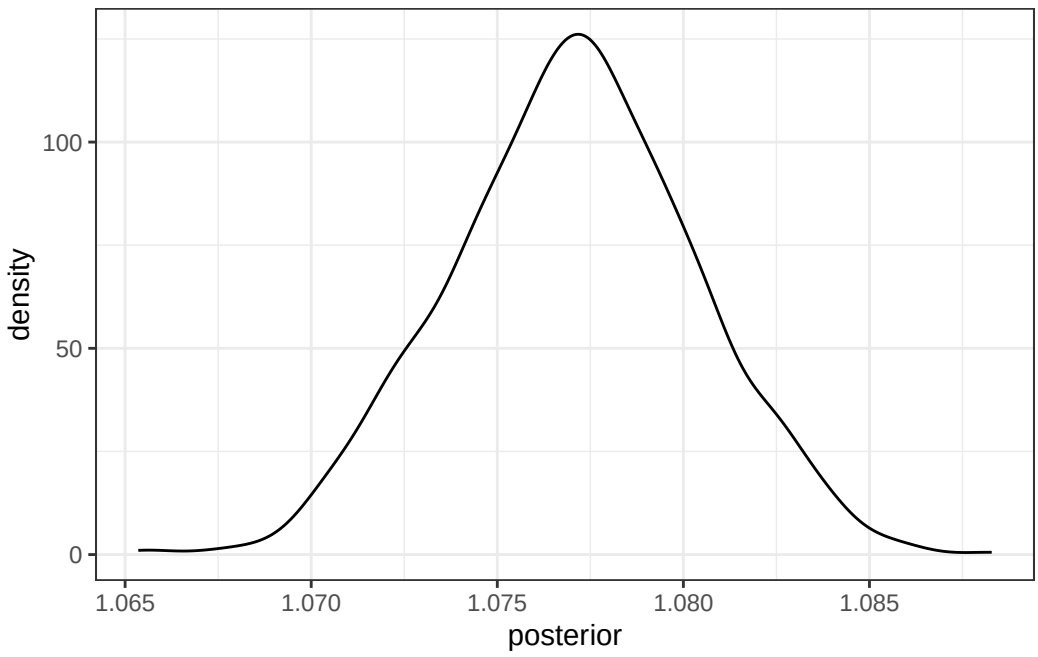
```
              75%     97.5%     n_eff      Rhat
beta[1] 1.1645491 1.1681526 1081.8199 0.9972360
beta[2] 0.1455863 0.1491984 1134.1163 0.9986568
sigma   0.1701874 0.1725273  336.8921 1.0084718
```

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.

The estimate of expected birthweight of a baby who born at a gestational age of 37 weeks has a mean weight of 2.93.

```
posterior_samples <- extract(mod1)
b1=posterior_samples$beta[,1]
b2=posterior_samples$beta[,2]
e=posterior_samples$sigma

posterior=b1+b2*(log(37) - mean(log(ds$gest)))/sd(log(ds$gest))

ggplot(posterior|>as.data.frame(),(aes(x=posterior))) +
  geom_density() + theme_bw()
```



```
exp(mean(posterior))
```

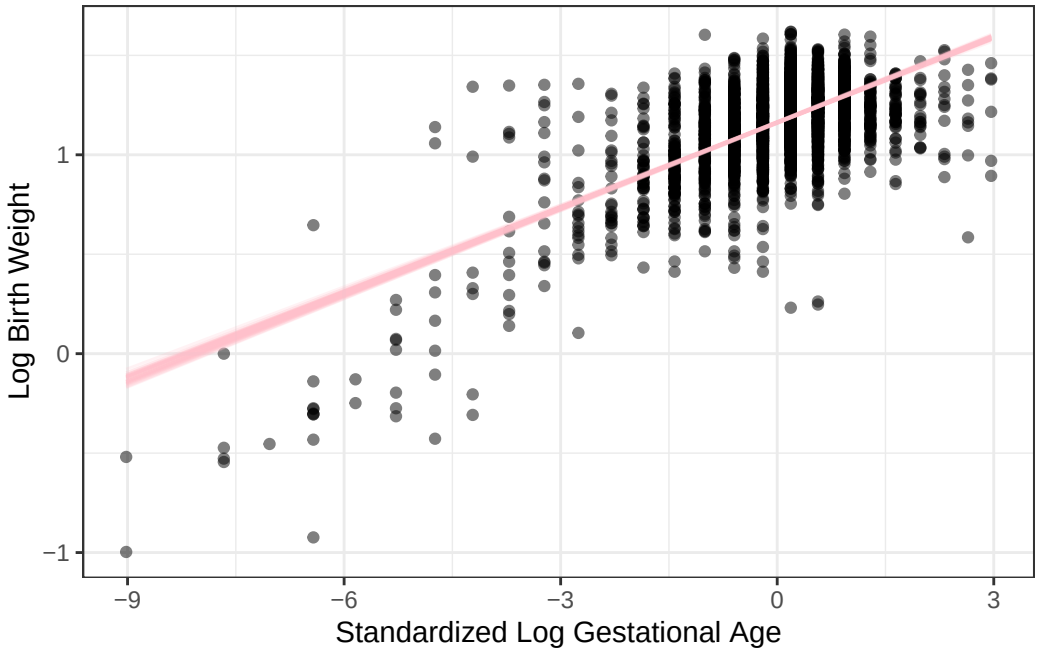
```
[1] 2.935993
```

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.

Below is a scatter plot showing the underlying data (ddenoted in black) and 50 posterior draws of the linear predictor (denoted in pink).

```
gest_standardized=(log(ds$gest) - mean(log(ds$gest)))/sd(log(ds$gest))
base_plot <- ggplot(ds, aes(x = gest_standardized, y = log_weight)) +
  geom_point(alpha = 0.5, color = "black") +
  theme_bw() +
  labs(x = "Standardized Log Gestational Age", y = "Log Birth Weight")

linear_preds <- expand.grid(gest_standardized = gest_standardized, iteration = 1:50)
linear_preds$linear_predictor <- with(linear_preds, b1[iteration] + b2[iteration] * gest_s

base_plot + geom_line(data = linear_preds, aes(x = gest_standardized, y = linear_predictor
```



Question 5

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

```
ds$log_weight <- log(ds$birthweight)
ds$log_gest_c <- (log(ds$gest) - mean(log(ds$gest)))/sd(log(ds$gest))
ds$preterm2<-ifelse(ds$preterm=="Y",1,0)
# put into a list
stan_data <- list(N = nrow(ds),
                  log_weight = ds$log_weight,
                  log_gest = ds$log_gest_c,
                  preterm2=ds$preterm2
                )
```

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

The summary of the results are shown below. In addition,the coefficient estimated in the interaction term beta3 indicates the effect of log of gestational age on log of the weight of the baby with the status of premature delivery. Given beta3=0.1098, it suggests that the effect of gestational age on birth weight is stronger for preterm birth than for non-preterm birth by 0.1098, which mean for every one unit increase in the log of gestational age, the expected increase in the log of weights of the baby is increased by 0.1098 units if the baby is perterm.

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

```
            mean      se_mean          sd      2.5%       25%       50%
beta[1] 1.1702705 7.343734e-05 0.002554424 1.1653060 1.1685569 1.1702246
beta[2] 0.1031413 1.172041e-04 0.003583603 0.0966312 0.1007832 0.1030286
beta[3] 0.1098036 1.730653e-04 0.004731409 0.1003088 0.1066045 0.1099198
sigma   0.1622125 9.060211e-05 0.001826036 0.1588462 0.1609954 0.1621489
             75%     97.5%     n_eff      Rhat
beta[1] 1.1720099 1.1750742 1209.9075 0.9987015
beta[2] 0.1056051 0.1106521  934.8763 0.9999060
beta[3] 0.1131088 0.1186360  747.4132 1.0029508
sigma   0.1634810 0.1659611  406.2019 1.0024491
```

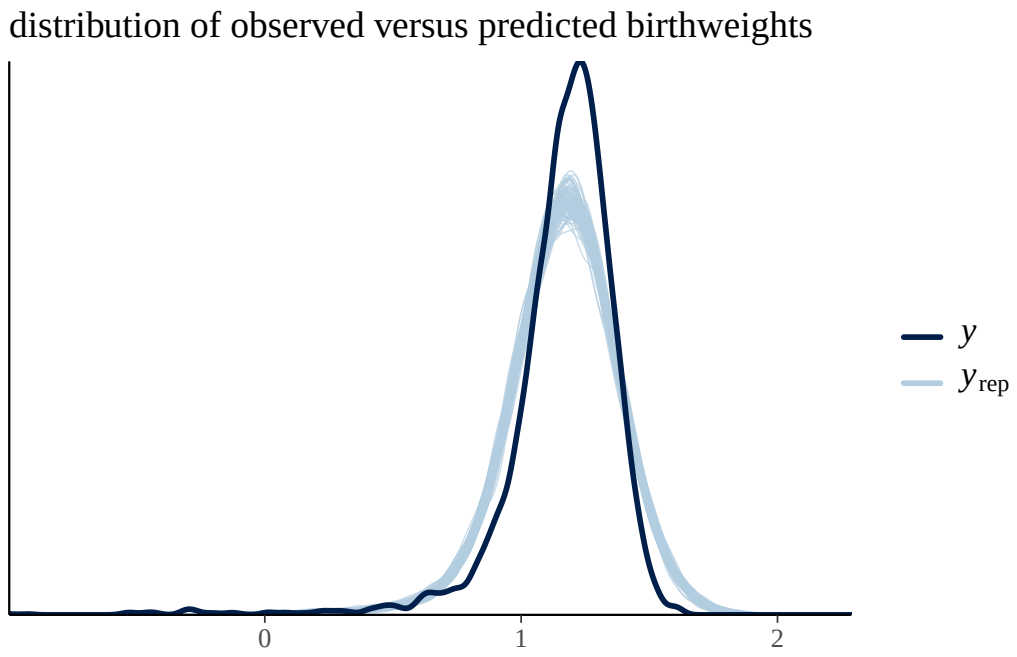
PPCs

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
```

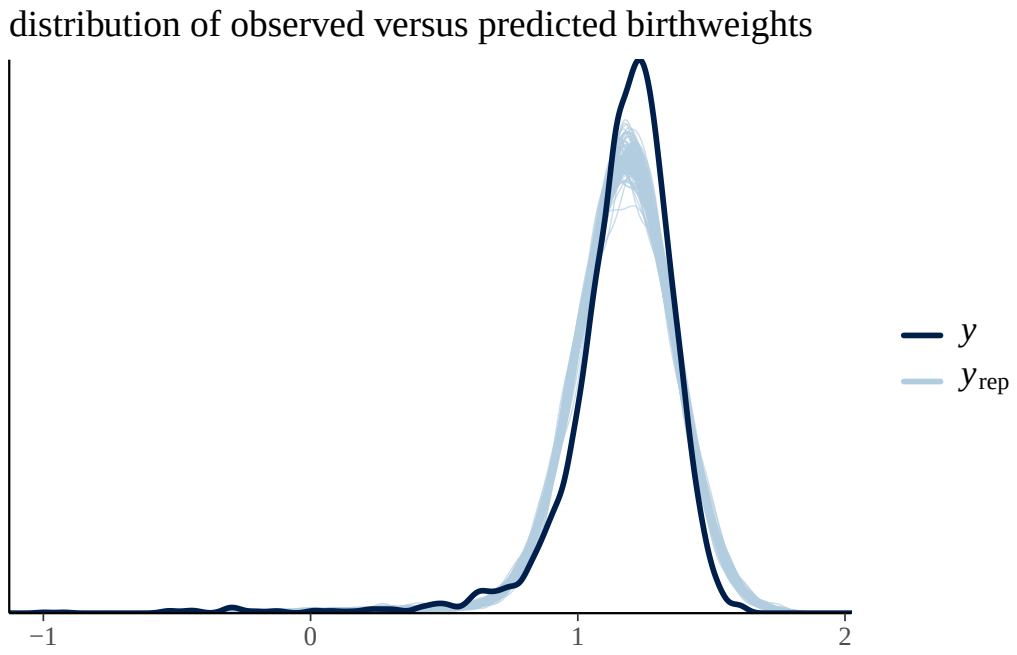
```
samp100 <- sample(nrow(yrep1), 100)
ppc_dens_overlay(y, yrep1[samp100, ])  + ggtitle("distribution of observed versus predicte
```



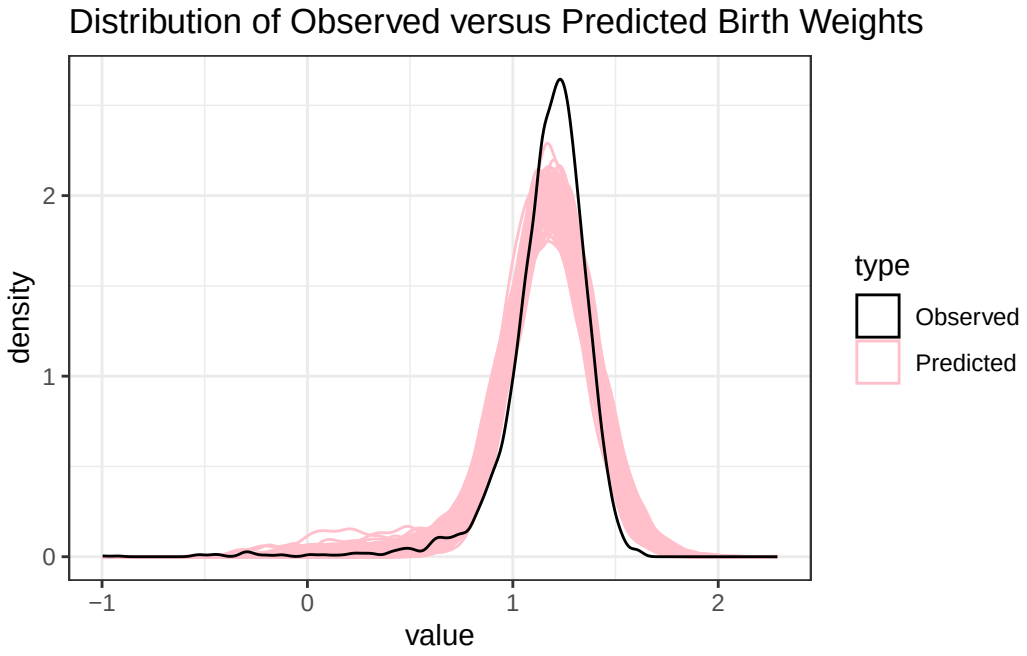
Question 6

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`)

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




```r
post_y <- yrep1[samp100,]

plot_data <- data.frame(
  value = c(as.vector(post_y), y),
  type = rep(c(rep("Predicted", nrow(post_y)), "Observed"), each = length(y)),
  sample = rep(0:nrow(post_y), each = length(y))
)

# Plot using geom_density
ggplot(plot_data, aes(x = value, color = type, group = interaction(type, sample))) +
  geom_density(alpha = 0.5) +
  scale_color_manual(values = c("black", "pink")) +
  scale_size_manual(values = c(10, 1)) +
  ggtitle("Distribution of Observed versus Predicted Birth Weights")+theme_bw()
```

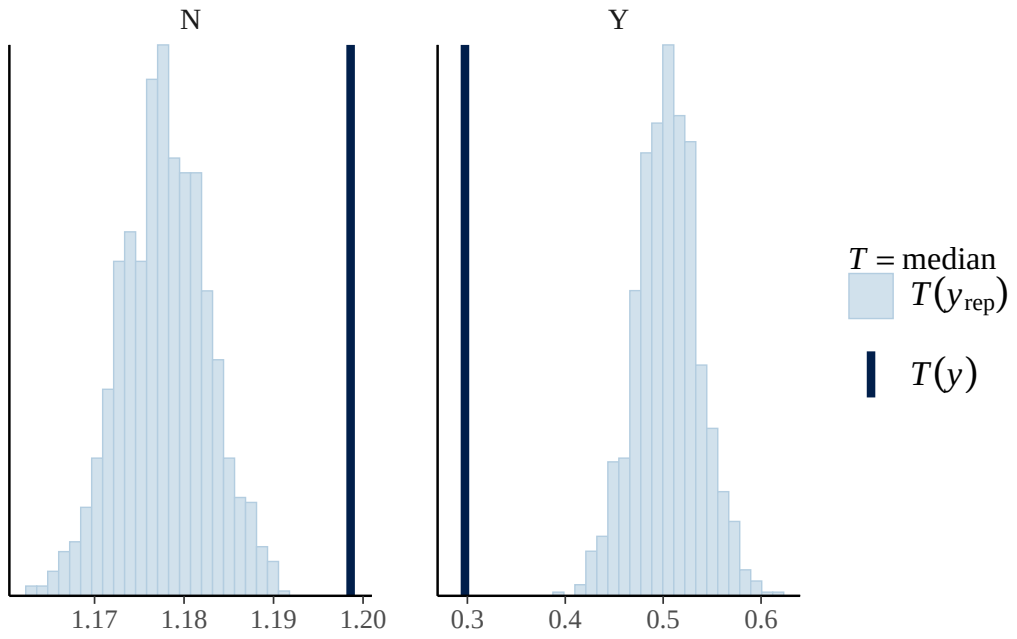


Test statistics

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')
```



Question 7

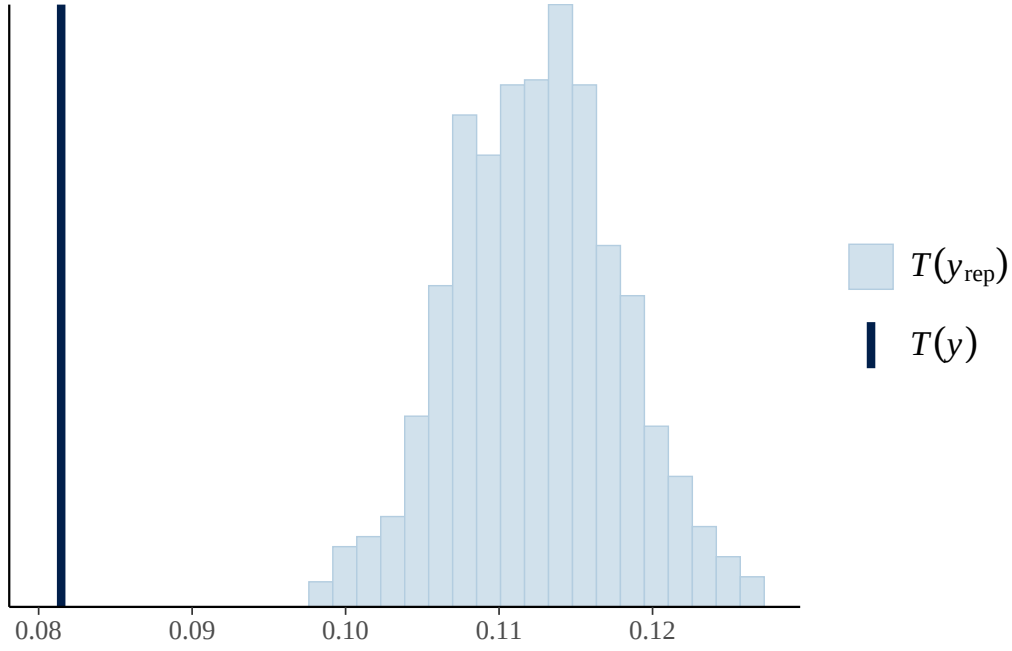
Use a test statistic of the proportion of births under 2.5kg. Calculate the test statistic for the data, and the posterior predictive samples for both models, and plot the comparison (one plot per model).

The test statistics of the data is around 0.0814.

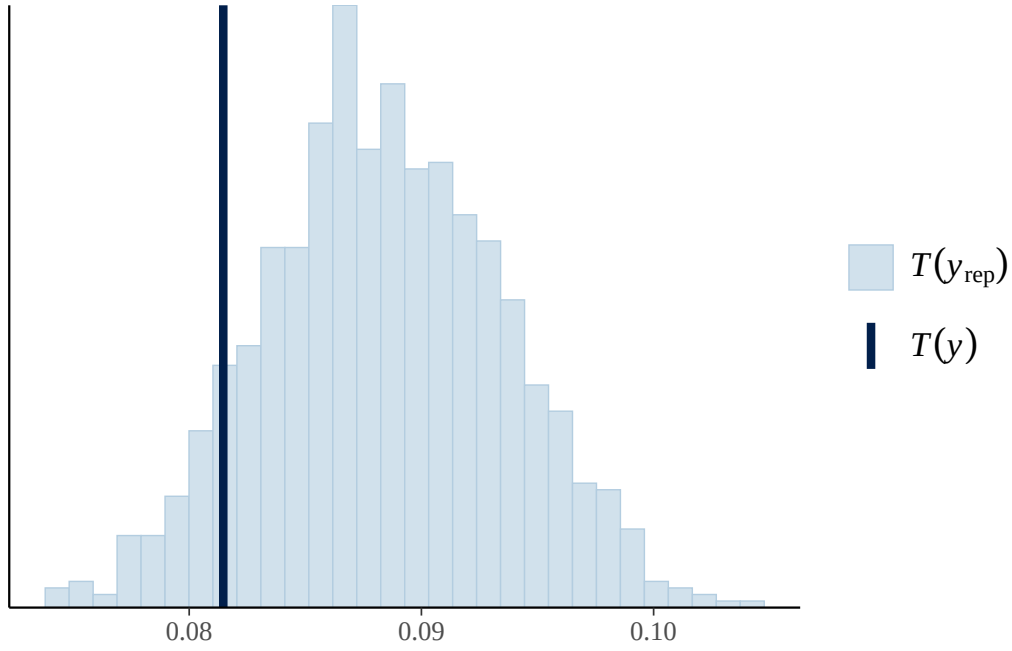
```
sum(ds$log_weight<log(2.5))/length(ds$log_weight)
```

```
[1] 0.08146799
```

```
ppc_stat(ds$log_weight, yrep1, stat = function(y) sum(y < log(2.5)) / length(y))
```




```
ppc_stat(ds$log_weight, yrep2, stat = function(y) sum(y < log(2.5)) / length(y))
```



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  72.4
p_loo         9.2   1.3
looic     -2754.8 144.9
------
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.

The loo estimate of elpd is computed for model2. The expected elpd for a new dataset is 1530.4 with a standard error of 71.9. `p_loo` represents the estimated effect number parameter. It is evaluated to be 10.2 for model 2, which is higher than that of model 1(9.2). The `looic` stands for LOO information citerion, it is estimated to be -3060.8 for model 2. Compared with model 1 which has a looic of -2754.8, model 2 has a lower looic, indicating a better fit of the model. It appears that the model 2 have a better predictive performance compared with model 1.

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

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

         Estimate    SE
elpd_loo   1530.4  71.9
p_loo        10.2   1.4
looic     -3060.8 143.8
------
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.
```

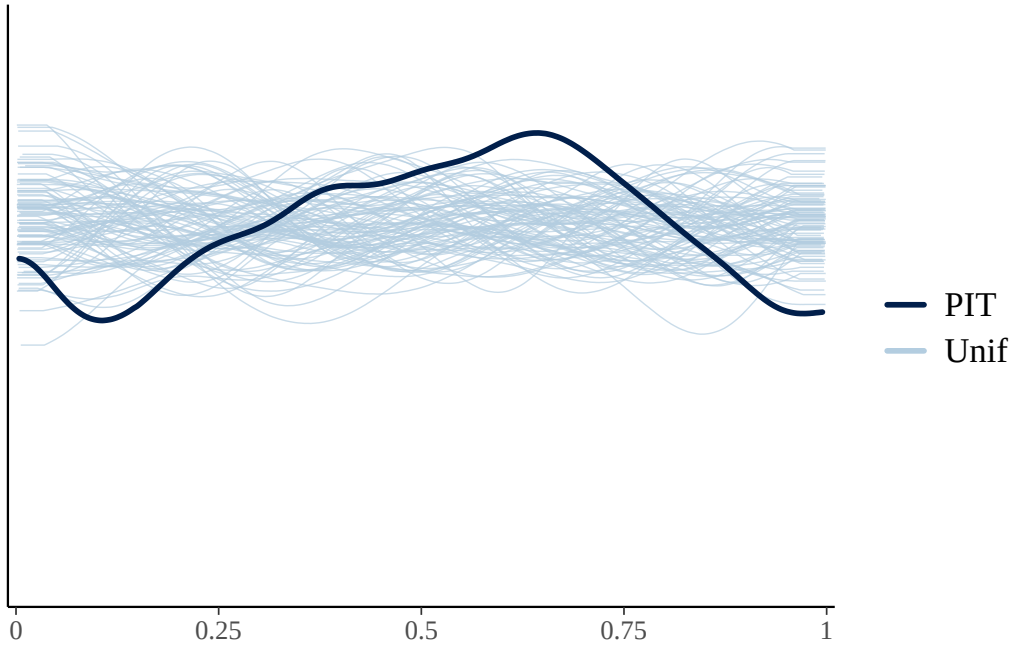
The output below shows the difference in the expected log predictive density (elpd) between model 1 and model 2. With model 2 being the reference, model 1 has a elpd 153 unit lower than that of model 2. Since the elpd of model 2 is higher, model 2 may have a better performance in terms of predictive performance.

```r
a=list(loo1,loo2)
loo_compare(a)
```

```
       elpd_diff se_diff
model2    0.0       0.0
model1 -153.0      35.3
```

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)

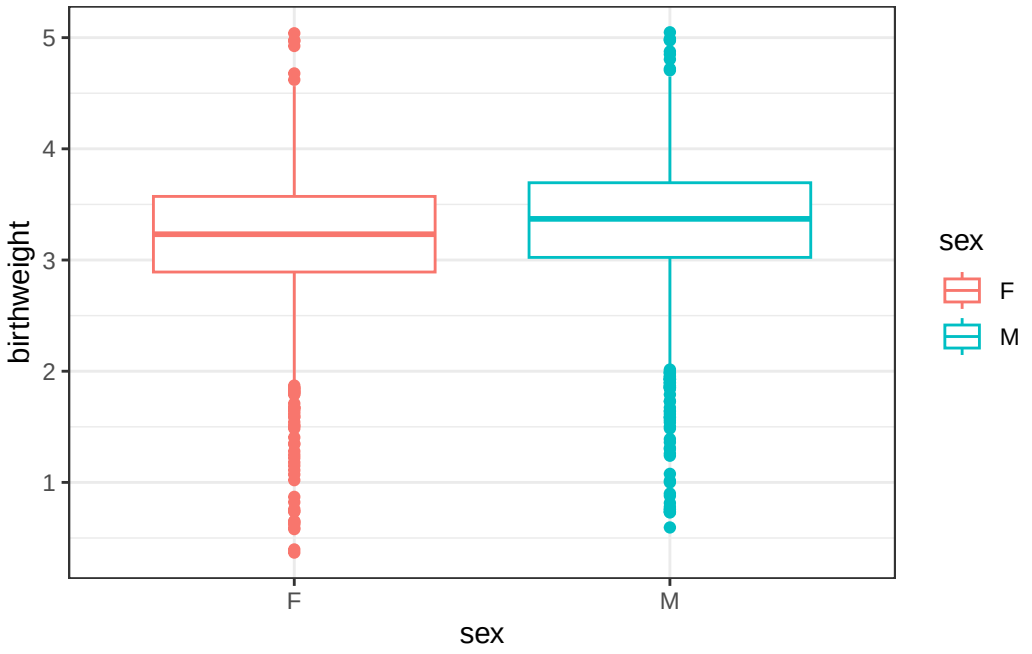
Create your own PIT histogram "from scratch" for Model 2.

Question 9

Motivation

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.

```
ggplot(data=ds,aes(x =sex,y=birthweight,color=sex))+geom_boxplot()+theme_bw()
```




```
#observations with `meduc` == 9 (unkown) are filtered out
```

Model fitting

From the plot above, we observe that the mean weight of male babies seems slightly higher than that of female babies. Therefore, the new covariate I would like to include is sex. Let's fit the model and report the model summary.

Model 3 is specified as follows

$$\log(y_i) \sim N(\beta_1 + \beta_2 \log(x_i) + \beta_2 z_i + \beta_3 \log(x_i) z_i + \beta_4 s_i, \sigma^2)$$

- 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)
- s_i is sex (0 or 1, if the baby is male)

```
ds$log_weight <- log(ds$birthweight)
ds$log_gest_c <- (log(ds$gest) - mean(log(ds$gest)))/sd(log(ds$gest))
ds$preterm2<-ifelse(ds$preterm=="Y",1,0)
ds$sex2<-ifelse(ds$sex=="M",1,0)
# put into a list
```

```r
stan_data_m3 <- list(N = nrow(ds),
                     log_weight = ds$log_weight,
                     log_gest = ds$log_gest_c,
                     preterm2=ds$preterm2,
                     sex2=ds$sex2
                   )

mod3 <- stan(data = stan_data_m3,
             file = here("labs/simple_weight3.stan"),
             iter = 500,
             seed = 243)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEI
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHeac
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEige
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEige
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Co
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Co
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHeac
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEige
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/Core:9
#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 0.001593 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 15.93 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: 1.192 seconds (Warm-up)
Chain 1:                1.387 seconds (Sampling)
Chain 1:                2.579 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000523 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 5.23 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 2: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 2: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 2: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 2: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 2: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 2: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 2: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 2: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 2: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 2: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 2: Iteration: 500 / 500 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 1.248 seconds (Warm-up)
Chain 2:                1.149 seconds (Sampling)
Chain 2:                2.397 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
```

```
Chain 3:
Chain 3: Gradient evaluation took 0.000745 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 7.45 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: 1.4 seconds (Warm-up)
Chain 3:                1.376 seconds (Sampling)
Chain 3:                2.776 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000715 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 7.15 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: 1.386 seconds (Warm-up)
Chain 4:                1.265 seconds (Sampling)
Chain 4:                2.651 seconds (Total)
Chain 4:
```

Fomr the summary below, we observe that the sex and birthweight are positively related with a coefficient of 0.043, which is small comapared to the effect of gestational ages.

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

```
             mean      se_mean          sd       2.5%        25%        50%
beta[1] 1.14879752 1.281845e-04 0.003599608 1.14204656 1.14638338 1.14894487
beta[2] 0.10386140 1.180119e-04 0.003454793 0.09763900 0.10130369 0.10376230
beta[3] 0.10898341 1.651399e-04 0.004807094 0.10031356 0.10550265 0.10891561
beta[4] 0.04288637 1.749227e-04 0.005080441 0.03249019 0.03973901 0.04294361
sigma   0.16084756 8.018451e-05 0.001885929 0.15712413 0.15950679 0.16074868
              75%      97.5%    n_eff      Rhat
beta[1] 1.15115402 1.15575581 788.5679 0.9975989
beta[2] 0.10627063 0.11056566 857.0232 0.9988225
beta[3] 0.11257986 0.11818184 847.3465 0.9989389
beta[4] 0.04625138 0.05250889 843.5492 0.9987150
sigma   0.16212276 0.16474890 553.1840 1.0049245
```

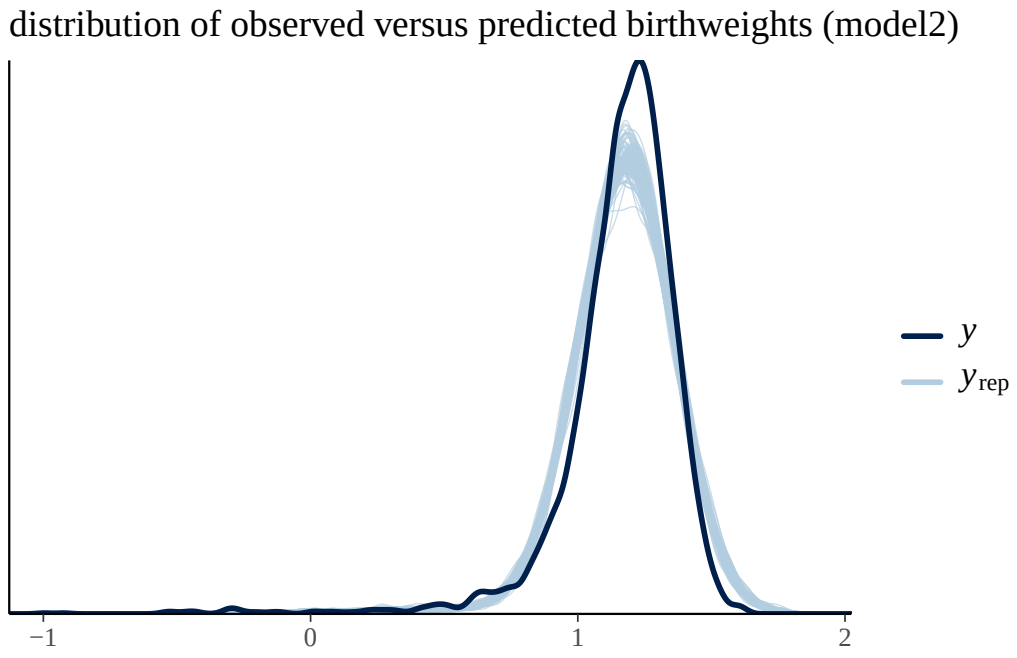
Model Checking

Next, let's do PPCs.

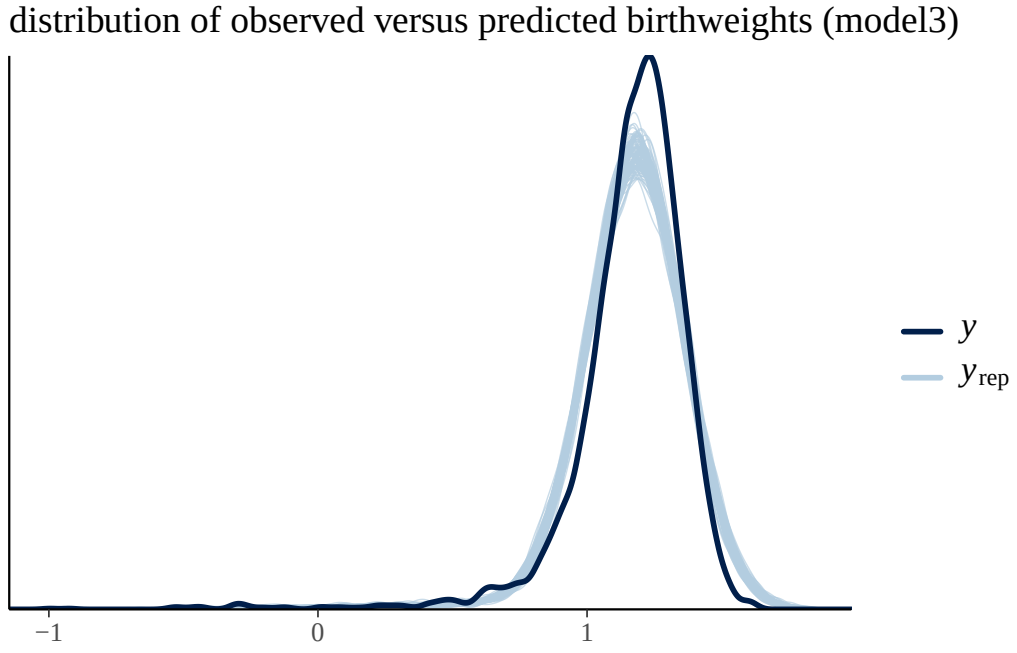
```
set.seed(1856)
y <- ds$log_weight
yrep3 <- extract(mod3)[["log_weight_rep"]]
samp100 <- sample(nrow(yrep3), 100)
```

Below are the overlayed density plots of model2 and model3. They compares the empirical distribution of the data y to the distributions of simulated/replicated data yrep from the posterior predictive distribution for model 2 and 3. Model 3 roughly have a similar empirical distribution with model 2.

```
par(mfrow = c(1, 2))
ppc_dens_overlay(y, yrep2[samp100, ])  + ggtitle("distribution of observed versus predicte
```




```
ppc_dens_overlay(y, yrep3[samp100, ])  + ggtitle("distribution of observed versus predicte
```



We then further compare hte models on their predictive powers. We observe that model 3 has the highest elpd, indicating model3 has better predictive performance comparing with model

1 and model 2. Model 3 is also more complex than model 1 and 2.

```
loglik3 <- extract(mod3)[["log_lik"]]
loo3 <- loo(loglik3, save_psis = TRUE)
loo3
```

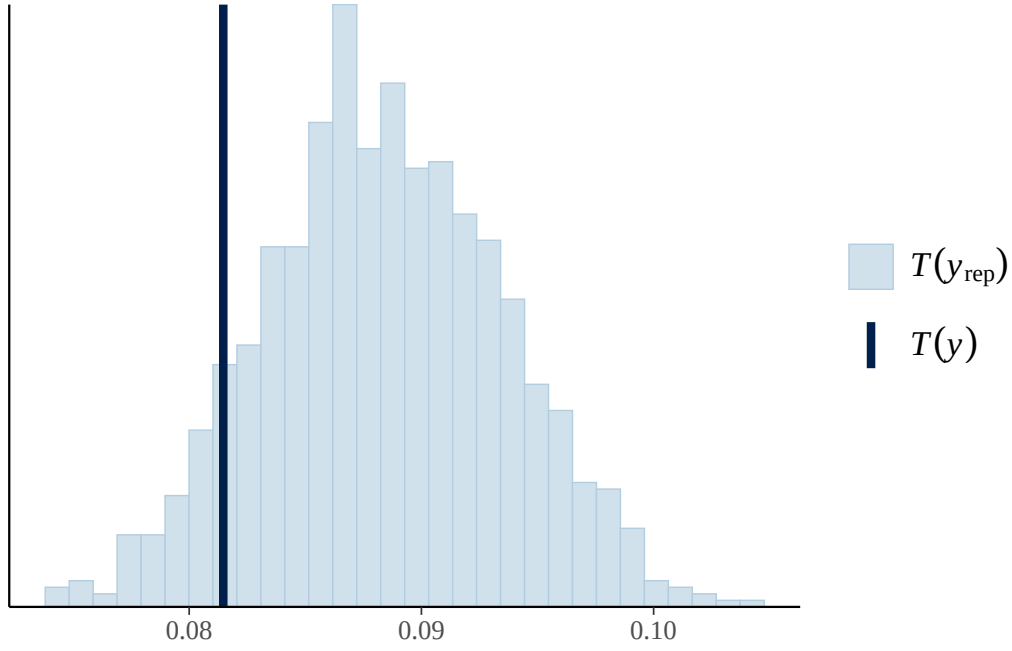
```
Computed from 1000 by 3842 log-likelihood matrix

         Estimate    SE
elpd_loo   1563.0  72.2
p_loo        11.4   1.4
looic     -3126.1 144.4
------
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.
```

```
b=list(loo1,loo2,loo3)
loo_compare(b)
```

```
       elpd_diff se_diff
model3    0.0       0.0
model2  -32.6       8.2
model1 -185.7      36.2
```

We also plot the test statistic specified in question 7 to compare model2 and model3.

```
ppc_stat(ds$log_weight, yrep2, stat = function(y) sum(y < log(2.5)) / length(y))
```




```
ppc_stat(ds$log_weight, yrep3, stat = function(y) sum(y < log(2.5)) / length(y))
```